BASIC η απλη γλωσσα των υπολογιστων

τό νά ἀγοράσει κανείς ἕνα ὑπολογιστή γιά δική του χρήση ἔχει γίνει πολύ δύσκολο, διότι πολλοί κυκλοφοροῦν στήν ἀγορά. Οἱ ὑπολογιστές ἔχουν γίνει σήμερα πολλοί ἀπαραίτητοι καί ὁ ἀριθμός τῶν ἐργασιῶν πού μποροῦν νά κάνουν εἶναι φοβερά μεγάλος. Μερικές ἀπό τίς δουλειές πού μπορεῖ νά κάνει σήμερα ἕνας ὑπολογιστής εἶναι:

- *Δημιουργία συστήματος οἰκονομίας ἐνέργειας στό σπίτι μέ τόν ἔλεγχο ἀπό τόν ὑπολογιστή τοῦ καλοριφέρ ἤ αἴρ - κοντίσιον.*
- *Αὐτόματο ἄνοιγμα καί κλείσιμο παραθύρων.*
- *Γράψιμο φορολογικῶν δηλώσεων.*
- *Προγραμματισμός δίαιτας: πόσα κιλά ἐπιθυμεῖ κάποιος νά χάσει, σέ πόσο χρόνο θέλει νά τά χάσει.*
- *Ὑπολογισμός προϋπολογισμῶν, ἰσολογισμῶν, κ.ἄ.*
- *Μετατροπή σέ σύστημα συναγερμοῦ.*
- *Ρύθμιση φώτων (ἀναβόσβησμα).*
- *Καταγραφή ἀριθμῶν τηλεφώνου κ.ἄ.*

Ὁ ὑπολογιστής εἶναι τό μηχάνημα τοῦ μέλλοντος καί ἀργά ἤ γρήγορα ὅλοι πρέπει νά μάθουν νά τόν χειρίζονται. Γιά νά μπορεῖ ὅμως κάποιος νά δώσει τίς διάφορες ἐντολές στόν ὑπολογιστή, θά πρέπει νά ἐπικοινωνήσει μαζί του, δηλ. θά πρέπει νά «μιλάει» τήν ἴδια «γλώσσα» μέ τόν ὑπολογιστή. Ὑπάρχουν διάφορες γλῶσσες. Μεταξύ αὐτῶν μιά ἀπό τίς σπουδαιότερες εἶναι τό "BASIC" σύστημα. Πρόκειται γιά πολύ εὔκολη γλώσσα προγραμματισμοῦ καί προορίζεται γι' αὐτούς πού οὔτε ξέρουν οὔτε ἔχουν τήν πρόθεση νά μάθουν πῶς λειτουργεῖ ὁ ὑπολογιστής τους. Δημιουργήθηκε στό Κολλέγιο Dartmouth τῆς Ἀμερικῆς στίς ἀρχές τοῦ 1960 καί ἀπό τότε ἔγινε μιά ἀπό τίς σπουδαιότερες γλῶσσες προγραμματισμοῦ. Τό ὄνομα προέρχεται ἀπό τίς λέξεις (Beginners All - purpose Symbolic Instruction Code), πού σημαίνει: συμβολικός κώδικας ὁδηγιῶν κάθε χρήσεως γιά ἀρχάριους.

Ἡ BASIC, μπορεῖ νά εἶναι ἁπλή, ἀλλά εἶναι πολύ χρήσιμη καί κυρίως ἐξυπηρετεῖ τούς ἀρχάριους πού ἔχουν λίγες γνώσεις γιά τή λειτουργία τῶν ὑπολογιστῶν.

Ἡ γλώσσα χρησιμοποιεῖ ἀγγλικές προτάσεις καί διαταγές. Γιά παράδειγμα ἄς δοῦμε πῶς θά γραφόταν μιά πρόσθεση γιά νά ἐκτελεσθεῖ ἀπό τόν ὑπολογιστή καί συγκεκριμένα ἡ πρόσθεση τῶν ἀριθμῶν 2 καί 3:

```
10A = 2 + 3
20 PRINT A
30 END
```

Οἱ ἀριθμοί 10, 20, 30 ὀνομάζονται ἀριθμοί γραμμῶν καί δείχνουν μέ ποιά σειρά θά ἐκτελεσθοῦν οἱ πράξεις. Οἱ ἀριθμοί αὐτοί χρησιμοποιοῦνται καί σά σημεῖο ἀναφορᾶς γιά ἄλλες πράξεις τοῦ BASIC. Ἡ πρώτη πρόταση δίνει στή μεταβλητή Α, τήν τιμή τοῦ ἀθροίσματος 2 καί 3. Ἡ πρόταση "PRINT A" γράφει τό ἀποτέλεσμα ἤ σέ display ἤ σέ γραφομηχανή, ἄν ὑπάρχει. Ἡ τελευταία πρόταση λέει στόν ὑπολογιστή ὅτι τό πρόγραμμα τελείωσε καί νά σταματήσει τίς πράξεις. Τό πρόγραμμα γράφεται στό πληκτρολόγιο μιά γραμμή κάθε φορά. Γιά νά ἐκτελεστεῖ τό πρόγραμμα, τυπώνεται ἡ λέξη "RUN" καί πατιέται εἰδικό κουμπί τοῦ πληκτρολογίου.

Ὑπάρχουν πολλές κατηγορίες τῆς BASIC (μερικές δωδεκάδες): ἡ ἁπλή καί ἡ προχωρημένη (EXTENDED) BASIC. Ἡ δεύτερη δίνει περισσότερα χαρακτηριστικά καί ἐντολές. Ἀκόμα καί στόν τρόπο πού εἶναι γραμμένο ἕνα πρόγραμμα ὑπάρχουν σημαντικές διαφορές. Ἡ χρήση ἄνω τελειῶν, τελειῶν, κομμάτων, εἰσαγωγικῶν, κ.ἄ, λέγεται σύνταξη (syntax).

Ἄλλες κατηγορίες τῆς BASIC εἶναι εὐαίσθητες καί ἄλλες ὄχι, μέ τή χρήση τῆς σύνταξης. Πρέπει νά ξέρει κανείς ἄν ὁ ὑπολογιστής του χρησιμοποιεῖ σύνταξη ἤ ὄχι. Ὑπάρχουν κι ἄλλες «γλῶσσες», ἀλλά καμιά δέν εἶναι τόσο εὔχρηστη ὅσο ἡ BASIC, διότι χρησιμοποιοῦν σύμβολα πού ἀπαιτοῦν εἰδικά πληκτρολόγια, ἐνῶ τά σύμβολα τῆς BASIC βρίσκονται σέ ὁποιοδήποτε κοινό πληκτρολόγιο.

Κανένας ὑπολογιστής δέν εἶναι κατασκευασμένος νά «καταλαβαίνει» ἀμέσως τό BASIC. Ἡ BASIC θά πρέπει νά μεταφράζεται στή «γλώσσα» τοῦ ὑπολογιστῆ. Τό νά ἀποθηκευτεῖ ἕνα μεταφραστικό ὅμως πρόγραμμα θά ἐλαττώσει τίς ὑπάρχουσες μνῆμες. Ἄν πάλι μποῦν λίγα «μεταφραστικά» ὁ ὑπολογιστής δέν θά ἔχει ὅλες τίς δυνατότητες πού θά ἔπρεπε γιά νά μεταφράσει ἕνα πρόγραμμα. Ἡ «μετάφραση» μπορεῖ νά γίνει μέ δύο διαφορετικούς τρόπους.

Ὁ πιό γρήγορος τρόπος εἶναι νά μεταφραστεῖ ὁλόκληρο τό πρόγραμμα πρῶτα καί ἐν συνεχεία νά ἀρχίσει νά ἐκτελεῖται. Ἕνα μεταφραστικό πρόγραμμα πού ἀκολουθεῖ αὐτή τή διαδικασία λέγεται "compiler" (συλλέκτης - συντάκτης).

Ὁ δεύτερος τρόπος εἶναι νά μεταφράζεται καί ἐκτελεῖται τό πρόγραμμα γραμμή πρός γραμμή (σταδιακά). Ἕνα μεταφραστικό πρόγραμμα πού ἀκολουθεῖ αὐτή τή διαδικασία λέγεται interpreter «μεταφραστής». Ἕνας «μεταφραστής» εἶναι ἄθροισμα ἀπό πολλά μικρά ξεχωριστά προγράμματα.

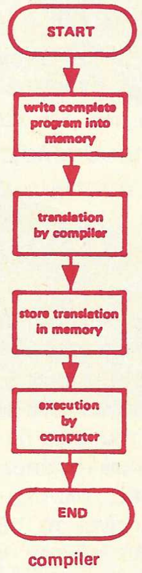
Ἡ διαδικασία λειτουργίας τοῦ compiler εἶναι ἡ ἑξῆς: ΑΡΧΗ → ΕΓΓΡΑΦΗ ΟΛΟΥ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΤΗ ΜΝΗΜΗ → μετάφραση ἀπό τόν compiler → ἀποθήκευση τῆς μετάφρασης στή μνήμη → ἐκτέλεση ἀπό τόν ὑπολογιστή → τέλος.

Ἡ διαδικασία λειτουργίας τοῦ interpreter εἶναι ἡ ἑξῆς: ΑΡΧΗ → ΕΓΓΡΑΦΗ ΟΛΟΥ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΤΗ ΜΝΗΜΗ → μετάφραση τῆς πρώτης (ἑπόμενης) γραμμῆς ἀπό τό μεταφραστή → ἐκτέλεση ἀπό τόν ὑπολογιστή → τέλος τοῦ προγράμματος; ἄν ναί, τέλος, ἄν ὄχι μετάφραση τῆς ἑπόμενης γραμμῆς → ἐκτέλεση ἀπό τόν ὑπολογιστή, κ.λ.π. μέχρι τό τέλος τοῦ προγράμματος.

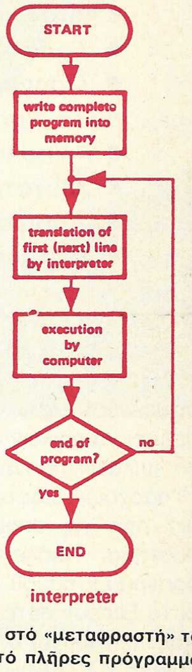
Πλεονεκτήματα τοῦ μεταφραστῆ

Τό σύστημα παρουσιάζει πλεονεκτήματα ὅταν πρόκειται νά ἐκτελεστοῦν μικρά προγράμματα κατ' εὐθείαν ἀπό τό πληκτρολόγιο. Ὅ,τιδήποτε συντακτικά λάθη ἤ λάθη προγραμματισμοῦ, διορθώνονται ἀμέσως σ' αὐτή τήν περίπτωση.

1



2



Σχ. 1. Ὅταν ἕνα πρόγραμμα μπαίνει στό «μεταφραστή» τό πρῶτο βῆμα εἶναι νά μεταφραστεῖ τό πλῆρες πρόγραμμα σέ γλῶσσα μηχανῆς.

Σχ. 2. Ἕνας «interpreter» μεταφράζει τό πρόγραμμα γραμμή πρός γραμμή. Ἀφοῦ μεταφραστεῖ ἡ γραμμή καί ἐκτελεστεῖ, τότε μεταφράζεται ἡ ἑπόμενη.

Μειονεκτήματα τοῦ μεταφραστῆ

Στά μεγαλύτερα προγράμματα βασικό μειονέκτημα εἶναι ὅτι πιθανόν ἴδιες φράσεις νά μεταφράζονται πολλές φορές. Ὁ compiler χρειάζεται νά μεταφράσει κάθε φράση μιά φορά καί ἀνατρέχει στή μετάφραση ὅποτε χρειαστεῖ.

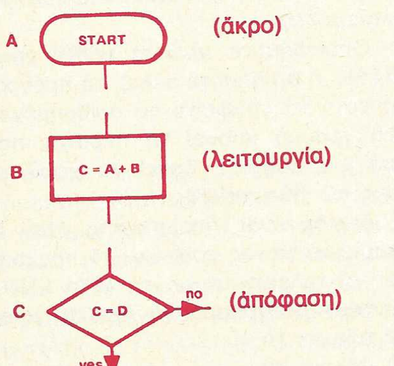
Ἔτσι γιά τά μεγάλα προγράμματα εἶναι προτιμότερο νά χρησιμοποιεῖται compiler, ἐνῶ γιά τά μικρά «μεταφραστής», χωρίς βέβαια αὐτό νά εἶναι κανόνας.

Πρίν ἀποφασίσει κανείς νά γράψει ἕνα πρόγραμμα σέ BASIC, θά πρέπει νά ἔχει μιά γενική εἰκόνα τοῦ «παρουσιαστικοῦ» ἑνός προγράμματος. Τό πρόγραμμα μπορεῖ νά παρομοιωθεῖ μέ ἕνα δρόμο πού ὁδηγεῖ ἀπό δοσμένες πληροφορίες σ᾽ ἕνα στόχο. Πολλές φορές μπορεῖ νά χρησιμοποιηθοῦν πολλοί δρόμοι καί ἀπαιτεῖται πείρα γιά νά διαλεχτεῖ ὁ καλλίτερος. Ὅταν ἀποφασιστεῖ ἡ διαδικασία σκιαγραφεῖται τό πρόγραμμα. Αὐτό τό διάγραμμα θά μεταφραστεῖ ἀκολούθως σέ BASIC.

Ἡ σπουδαιότητα τοῦ διαγράμματος εἶναι μεγάλη, διότι ἐξηγεῖ τήν κάθε διαδικασία τοῦ προγράμματος. Ἕνας τρίτος μπορεῖ νά καταλάβει ἕνα πρόγραμμα ἀπό ἕνα καλό διάγραμμα καί μόνο καί αὐτό καθοδηγεῖ τόν ἐκτελεστή γιά νά ἀκολουθήσει σωστό δρόμο.

Στά διαγράμματα αὐτά χρησιμοποιοῦνται εἰδικά σύμβολα. Τά πιό κοινά εἶναι σάν κι ἐκεῖνα πού σκιαγραφοῦν ἕνα πρόγραμμα (Σχ. 3).

3



Σχ. 3. Τά βασικά σύμβολα πού χρησιμοποιοῦνται σ᾽ ἕνα διάγραμμα σειρᾶς.

Τό πρῶτο στάδιο ὁποιουδήποτε σχεδιαγράμματος, εἶναι τό "start" (ἀρχή). Τό start (A) τοποθετεῖται μέσα στό σχῆμα πού φαίνεται καί μοιάζει μέ ἀκροδέκτη. Τό ἴδιο σύμβολο χρησιμοποιεῖται μέ τή λέξη "end" (τέλος) γιά νά δηλώσει τή λήξη ἑνός προγράμματος (E). Οἱ «λειτουργίες» ἑνός προγράμματος περιέχονται σέ ἕνα ὀρθογώνιο (B). Ἡ λειτουργία περιγράφεται ἀπό τίς φράσεις μέσα στό ὀρθογώνιο. Οἱ φράσεις πρέπει νά εἶναι σύντομες. Μόνο τά βήματα πού ἀπαιτοῦνται γιά κατανόηση τοῦ προγράμματος, πρέπει νά γράφονται.

Τά περισσότερα προγράμματα περιέχουν περισσότερο ἀπό μιά διαδοχή πράξεων. Ὁ ὑπολογιστής πιθανόν κάπου νά χρειαστεῖ νά πάρει ἀπόφαση. Οἱ ἀποφάσεις μπαίνουν σέ σχῆμα διαμαντιοῦ (C). Στό δοσμένο παράδειγμα συγκρίνονται οἱ τιμές τῶν C καί D. Ὁ ὑπολογιστής πρέπει νά ἀποφασίσει ἄν αὐτά τά δύο εἶναι ἴσα ἤ ὄχι καί ἡ μετέπειτα πορεία ἐξαρτᾶται ἀπό τήν ἀπάντηση "yes" ἤ "no" (ναί ἤ ὄχι). Μπορεῖ νά παρθεῖ ἀπόφαση καί μετά ἀπό τρεῖς συγκρίσεις ὅπως: if C>D, then (1), if C=D, then (2), if C<D, then (3), πού σημαίνει: ἄν C>D, τότε διαδικασία (1), ἄν C=D, τότε (2), ἄν C<D, τότε (3).

Αὐτό πού πρέπει νά τονιστεῖ σ᾽ αὐτό τό σημεῖο εἶναι ὅτι τό πρόγραμμα γράφεται **πάντοτε** ἀγγλικά, διότι οἱ ὑπολογιστές εἶναι προγραμματισμένοι νά καταλαβαίνουν μόνο αὐτή τή γλώσσα. Γι᾽ αὐτό καί τά παραδείγματα πού δίνουμε εἶναι γραμμένα στά Ἀγγλικά.

Συνεχίζοντας τήν ἐξήγηση τοῦ σκιαγράμματος φτάνουμε στό D. Πολλές φορές, κατά τήν ἐκτέλεση ἑνός προγράμματος, ἀπαιτεῖται ἡ γνώση ἐνδιάμεσων πράξεων ἤ θέλουμε νά ἐλέγξουμε ἐνδιάμεσες πράξεις γιά νά βεβαιωθοῦμε ὅτι τό πρόγραμμα προχωρεῖ καλά καί δέν χρειάζεται ἀνάκληση. Μέ τή λέξη "PRINT" (τύπωσε) μέσα στό παραλληλόγραμμο D διατάζουμε τόν ὑπολογιστή νά παρουσιάσει στό display τό ζητούμενο ἀποτέλεσμα. Τά τέσσερα αὐτά σύμβολα εἶναι ἀρκετά γιά ἕνα ἁπλό πρόγραμμα. Ἄν τό πρόγραμμα εἶναι πιό σύνθετο, πιθανό νά χρησιμοποιη-

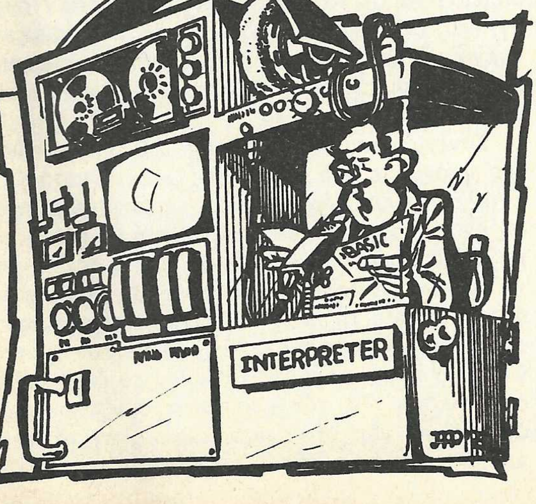
θοῦν καί ἄλλα σύμβολα. Ἕνα ἄλλο σύμβολο εἶναι ὁ «συνδετήρας» F, πού χρησιμοποιεῖται γιά νά ἑνώσει διάφορα ξεχωριστά μέρη ἑνός προγράμματος. Μ' αὐτά καί μόνο τά σύμβολα μπορεῖ κανείς νά σκιαγραφήσει πρόχειρα ἕνα πρόγραμμα.

Ὑπολογιστής τσέπης ἤ computer;

Ὑπάρχουν βασικές διαφορές ἀνάμεσα σ' αὐτά τά δύο πράγματα. Ἕνας φτηνός ὑπολογιστής τσέπης διαθέτει πληκτρολόγιο ἐνσωματωμένο, ἕνα ψηφιακό κύκλωμα καί τό display καί μπορεῖ νά κάνει τίς τέσσερις πράξεις τῆς ἀριθμητικῆς. Τό πληκτρολόγιο τοῦ ὑπολογιστῆ ἀντίθετα, εἶναι ξεχωριστό ἀπό τό ὑπόλοιπο μηχάνημα.

Οἱ ἐντολές μπαίνουν στό computer ἀπό ἕνα ἀκροδέκτη, πού πολλές φορές ἑνώνεται μέ σύρματα μέ τό computer. Ὁ ἀκροδέκτης αὐτός εἶναι ὁ συνδετικός κρίκος ἀνάμεσα στόν ἄνθρωπο καί τό μηχάνημα. Κάνει δυνατή τή συζήτηση ἀνάμεσα στούς δύο. Τό πληκτρολόγιο μοιάζει καί πολλές φορές δουλεύει ὅπως μιά γραφομηχανή, τυπώνοντας τίς ἐντολές σέ κοινό χαρτί. Τό πληκτρολόγιο συνοδεύεται καί ἀπό μιά ὀθόνη, τήν VDU.

Ἡ δεύτερη βασική διαφορά ἀνάμεσα στά δύο μηχανήματα εἶναι ὁ τρόπος μέ τόν ὁποῖο παρουσιάζονται τά ἀποτελέσματα. Ἕνας ὑπολογιστής τσέπης ἐμφανίζει μόνο ἀριθμούς, ἐνῶ ἕνας computer μπορεῖ νά προγραμματιστεῖ νά τυπώσει μέ ἀριθμούς καί λέξεις σέ χαρτί τό ἀποτέλεσμα ἤ νά τό τυπώσει ἐπάνω στή VDU.



Ὁ ὑπολογιστής, ἀνάλογα μέ τήν κατασκευή, εἶναι προγραμματισμένος νά δείξει ὅτι εἶναι ἕτοιμος νά ἀρχίσει νά δουλεύει ἐκθέτοντας στήν ὀθόνη τή λέξη "hello" (γειά σας). Τό τί πρέπει νά κάνει ὁ χειριστής γιά νά «ἑτοιμάσει» τόν ὑπολογιστή, περιγράφεται στό βιβλίο κατασκευῆς τοῦ ὑπολογιστῆ.

Δύο τέτοια παραδείγματα, ἀνάλογα μέ τό μεταφραστή πού χρησιμοποιεῖται, εἶναι ἡ ἔκθεση τῶν λέξεων:

```
ROM BASIC 1.0
READY
#

DCE TINY BASIC V1.0
OK
>
```

Ὁ πρῶτος τρόπος χρησιμοποιεῖται ἀπό τό Motorola M6800 μικροϋπολογιστή, ὁ δεύτερος ἀπό τόν 8080. Ὁ AMI S6800 χρησιμοποιεῖ τά σύμβολα: , ὁ NIBL τό >. Αὐτά τά σύμβολα λέγονται prompts καί δείχνουν ὅτι ὁ μεταφραστής εἶναι ἕτοιμος νά δεχθεῖ πληροφορίες ἀπό τό πληκτρολόγιο.

Ὁποιαδήποτε γλώσσα κι ἄν χρησιμοποιεῖται (BASIC ἤ ὁ,τιδήποτε ἄλλη), τό πρόγραμμα πρέπει πάντοτε νά γράφεται σέ ἀριθμημένες γραμμές. Μιά γραμμή μπορεῖ νά περιέχει περισσότερες ἀπό μία ὁδηγίες, ἀρκεῖ νά χωρίζονται μεταξύ τους μέ δύο τελεῖες (:).

Ἐπειδή ἕνας ὑπολογιστής ὅταν ἀρχίσει δέν σταματάει μόνος του, ὅλα τά προγράμματα πρέπει νά τελειώνουν μέ τή λέξη END. Ἔτσι ἕνα σύντομο πρόγραμμα θά ἔχει τή μορφή:

```
10 ὁδηγία 1η
20 ὁδηγία 2η
30 ὁδηγία 3η
35 ὁδηγία 4η : ὁδηγία 5η
40 ὁδηγία 6η
50 END (τέλος).
```

Μιά εὔλογη ἐρώτηση εἶναι, γιατί οἱ γραμμές ἀριθμοῦνται 10, 20, 30, κ.λ.π., γιατί δηλαδή παρουσιάζουν κενά μεταξύ τους; Στήν πράξη, ἡ πρώτη προσπάθεια τῆς καταγραφῆς ἑνός προγράμματος σπάνια εἶναι ὁλοκληρωμένη - παραβλέπονται σχεδόν πάντα ὁρισμένες ὁδηγίες.

Ἔτσι στά κενά θά μποῦν ὁδηγίες πού θά σκεφτεῖ ὁ χειριστής ἐκ τῶν ὑστέρων, χωρίς νά χρειασθεῖ νά ἀλλάξει ὅλο τό πρόγραμμα. Ἕνα

τέτοιο παράδειγμα εἶναι ἡ γραμμή 35. Ὁ ὑπολογιστής κάνει τίς λειτουργίες μέ τόν ἀριθμό τῆς σειρᾶς. Ἔτσι ἄν μετά τό 50 μπεῖ ἡ γραμμή 2, αὐτή θά ἐκτελεστεῖ ἀμέσως μετά τή γραμμή 1. Εἶναι δυνατό ἀκόμα νά τροποποιηθεῖ μιά γραμμή ἐκ τῶν ὑστέρων, ἀνατυπώνοντας τή γραμμή μέ τή σχετική τροποποίηση. Γιά νά διαγραφεῖ μία γραμμή, ἀρκεῖ νά γραφεῖ ὁ ἀριθμός της συνοδευόμενος ἀπό τή λέξη CR (CARIAGE RETURN). Στό τέλος κάθε γραμμῆς πρέπει νά μπαίνει τό CR, γιά νά εἰδοποιεῖ τόν ὑπολογιστή νά ἀλλάξει γραμμή.

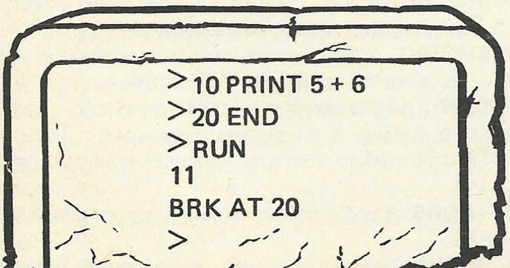
Προτάσεις

Ὅπως ὅλες οἱ γλῶσσες, ἔτσι καί ἡ BASIC χρησιμοποιεῖ λέξεις ἀγγλικές. Τό λεξιλόγιο εἶναι λίγο καί ἁπλό, ὥστε κι ἕνας ἀρχάριος τό μαθαίνει πολύ εὔκολα. Μιά λέξη εἶναι μιά ὁδηγία πού πρέπει νά κάνει ὁ ὑπολογιστής. Αὐτές οἱ ὁδηγίες λέγονται προτάσεις.

Γιά νά φανεῖ ἕνα ἀποτέλεσμα πρέπει νά χρησιμοποιηθεῖ ἡ λέξη PRINT (τύπωσε), ἀλλιῶς ὁ ὑπολογιστής θά ἀποθηκεύσει στή μνήμη τό ἀποτέλεσμα. Ἐάν πρόκειται νά γίνει ἕνας σύντομος ὑπολογισμός πού δέν θά τοποθετηθεῖ στή μνήμη, μπορεῖ νά μή χρησιμοποιηθοῦν ἀριθμήσεις γραμμῶν ὅπως:

```
> PRINT 5+6
11
>
```

Μετά τό πρῶτο "prompt" τυπώνεται ἡ λέξη PRINT καί ἀμέσως μετά ὁ ζητούμενος ὑπολογισμός. Ἐάν χρησιμοποιηθεῖ τό CR, στήν ἀμέσως ἑπόμενη γραμμή τυπώνεται τό ἀποτέλεσμα, καί τυπώνεται ἕνα ἄλλο "prompt" δείχνοντας ὅτι εἶναι ἕτοιμο γιά ἄλλη ἐντολή. Τό πρόγραμμα τῆς πάραπάνω ἐντολῆς θά τοποθετηθεῖ στή χρήση καί ἔχει ὡς ἑξῆς:



Στίς δύο πρῶτες γραμμές τυπώνεται τό πρόγραμμα, μετά παίρνει τήν ἐντολή RUN (τρέξε) καί δίνει τό ἀποτέλεσμα, ἄν μετά τό RUN πατηθεῖ τό CR. Ἡ διαφορά ἑνός ὑπολογισμοῦ ἀπό ἕνα πρόγραμμα εἶναι ὅτι ἄν ἀργότερα πατηθεῖ πάλι τό RUN, τό πρόγραμμα θά βγάλει τό 11, ἐνῶ ὁ ὑπολογισμός θά δείξει ἕνα ἐρωτηματικό (?) ἤ ἀποτέλεσμα ἀπό κάποιο ἄλλο πρόγραμμα. Τό BRK 20 εἶναι σύντομο γιά "BREAK AT 20" (τό πρόγραμμα περατοῦται στή γραμμή 20). Αὐτή ἡ ἔκφραση διαφέρει ἀπό ὑπολογιστή σέ ὑπολογιστή. Ἄλλοι μπορεῖ νά παρουσιάζουν τή λέξη READY. Ὁ ὅρος αὐτός ἀκολουθεῖται ἀπό ἕνα CR, ἕνα LF (μεταφορά στήν ἑπόμενη γραμμή) καί ἕνα σύμβολο prompt. Σ' ἕνα πρόγραμμα μπορεῖ νά χρησιμοποιηθοῦν πολλά PRINT, π.χ.

```
> 10 PRINT "5 + 6 ="
> 20 PRINT 5 + 6
> 30 END
> RUN
5 + 6 =
11
BRK AT 30
>
```

Βάζοντας τά εἰσαγωγικά ὑποχρεώνουμε νά τυπωθεῖ ἡ φράση «5+6» καί στήν ἑπόμενη γραμμή τό ἀποτέλεσμα. Θά ἦταν πιό ὡραῖο νά τυπωθεῖ ἡ φράση καί τό ἀποτέλεσμα σέ μιά γραμμή. Αὐτό πετυχαίνεται ὡς ἀκολούθως:

```
> 10 PRINT "5 + 6 =";
> 20 PRINT 5 + 6
> 30 PRINT
> 40 PRINT "5 + 6 ="; 5 + 6
> 50 END
> RUN
5 + 6 = 11

5 + 6 = 11
READY
>
```

Τό ἀποτέλεσμα τῆς ἄνω τελείας (;) εἶναι νά γραφεῖ ἡ ἑπόμενη ὁδηγία PRINT στήν ἴδια γραμμή. Ἡ ἄνω τελεία χρησιμοποιεῖται γιά νά ξεχωρίσει προτάσεις, ὅπως φαίνεται στή γραμμή 40, παράγοντας τό ἴδιο ἀποτέλεσμα ὅπως καί οἱ γραμμές

10 καί 20. Ένα ἀκόμα σημεῖο χρειάζεται διευκρίνιση στό προηγούμενο πρόγραμμα. Ἡ γραμμή 30 πιθανόν νά φαίνεται περιττή. Ὁ σκοπός της εἶναι νά ἀφήνεται μιά γραμμή κενή.

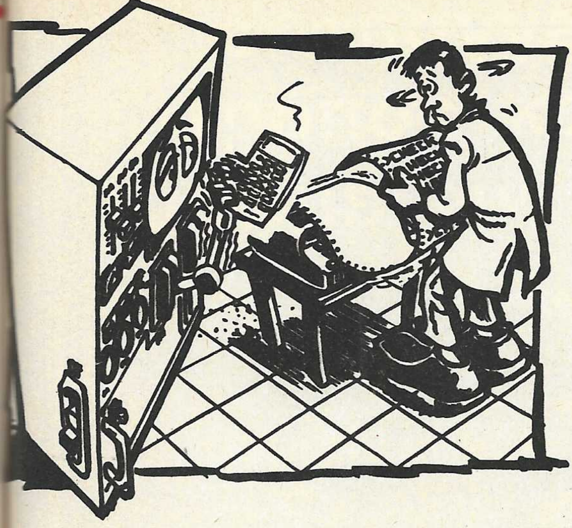
Ὑπάρχουν ὅμως μερικές ἀκόμα ἀναλύσεις προγραμμάτων πού πρέπει νά συζητηθοῦν καί νά δοθοῦν μετά καί ἄλλες δυνατότητες στήν πρόταση PRINT.

(ΣΥΝΕΧΙΖΕΤΑΙ)

Λεξικό ὅρων καί συμβόλων

Γιά νά μπορεῖ ὅμως κάποιος νά γράψει προγράμματα γενικά θά πρέπει νά ἔχει ὑπ' ὄψη του τά βασικά σύμβολα καί τήν ὁρολογία τῶν προγραμμάτων. Παραθέτουμε παρακάτω τά βασικότερα ἀπ' αὐτά:

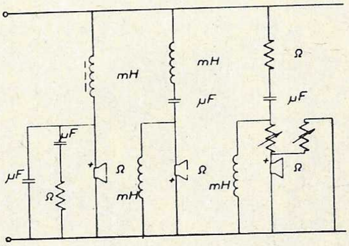
ACCUMULATOR (ΚΑΤΑΧΩΡΗΤΗΣ): Ἐκεῖ πού καταχωροῦνται λογικά ἤ ἀριθμητικά ἀποτελέσματα.
ASCII: Κώδικας γιά πληροφορίες. Συνήθως ὁ κώδικας εἶναι δυαδικός.
ASSEMBLER: Πρόγραμμα πού μετατρέπει γλώσσα προγραμματισμοῦ σέ μηχανική γλώσσα, καί ἐλέγχει ἄχρηστες ἤ ἀτελεῖς προτάσεις.
ASSEMBLY LANG: Ἀναπαράσταση προτάσεων στή μνήμη μέ σύμβολα.
ASYNCHRONOUS: (ἀσυγχρονισμός): λειτουργία πού εἰσάγει νέα διαδικασία μή προγραμματισμένη.
BASIC: Εὐκολόχρηστη ὑψηλῆς στάθμης γλώσσα.
BAUD: Συχνότητα μέ τήν ὁποία γίνεται ἀλλαγή γραμμῆς ἀνά δευτερόλεπτο.
BAUDOT CODE: Κώδικας 5 bit γιά ἀποθήκευση ἀλφαριθμητικῶν πληροφοριῶν.
BCD: Ἀντιστοιχία δεκαδικῶν μέ δυαδικούς ἀριθμούς.
BINARY: Δυαδικό σύστημα ἀρίθμησης. Οἱ ἀριθμοί πού χρησιμοποιοῦνται εἶναι τό 0,1.
BIT: Ἕνα δυαδικό ψηφίο.
BUFFER: Σύστημα πού ἀπομονώνει εὐαίσθητο τμῆμα συστήματος ἀπό τό ὑπόλοιπο σύστημα.
BUG: Λάθος πού ἀναγκάζει τό πρόγραμμα νά σταματήσει.
BUS: Οἱ ἐνδοσυνδέσεις συστήματος πού μεταφέρουν παράλληλες δυαδικές πληροφορίες.
BYTE: Σύνολα ἀπό bits - συνήθως 8.
COMPILER: Μεταφραστής πού μεταφράζει μιά γλώσσα σέ μηχανική πρίν ἀρχίσει ἡ ἐκτέλεση τοῦ προγράμματος.
CPU: Τό τμῆμα πού κάνει τούς ὑπολογισμούς
CUTS: Σύστημα πού ἀποθηκεύει πληροφορίες σέ κασσέτες ὑπό μορφή ἤχων πού ἀντιστοιχοῦν στό 0 ἤ 1.
DEBUG: Διόρθωση λαθῶν.
DMA: Ἀπ' εὐθείας μνήμη.
EPROM: Μνήμη πού σβήνεται (συνήθως μέ ὑπεριῶδες φῶς).
EXECUTE: Ἐκτέλεση σειρᾶς διαδικασιῶν.
EXECUTION TIME: Ἡ ὥρα πού χρειάζεται νά ἐκτελεσθεῖ μιά λειτουργία.
FIRMWARE: Πληροφορίες ἀποθηκευμένες στό ROM.
FLOPPY (DISK): Μαγική ἀποθήκευση πού γίνεται σέ εὔκαμπτους δίσκους.
FLOW CHART: Σχεδιάγραμμα τῆς λογικῆς διαδικασίας προγράμματος.
HARD COPY: Ἔξοδος τυπωμένη σέ χαρτί.
HARDWARE: Τά ἠλεκτρονικά καί μηχανικά μέρη συστήματος.
HEXADECIMAL: Δεκαεξαδικό σύστημα. Χρησιμοποιοῦνται οἱ ἀριθμοί 0 μέχρι 9 καί τά γράμματα A μέχρι F.
HIGH LEVEL LANGUAGE: Εὐκολόχρηστη γλώσσα πού ἀπαιτεῖ ὅμως μετατροπή σέ μηχανική πρίν χρησιμοποιηθεῖ.
INSTRUCTION: Τά bit πού τοποθετοῦνται γιά νά γίνει μιά λειτουργία.
INSTRUCTION SET: Ἡ σειρά τῶν instruction πού μπορεῖ νά ἐκτελέσει ἕνας ὑπολογιστής.
INTERFACE: Κύκλωμα πού συνδέει διάφορα τμήματα.
INTERPRETER: Μεταφραστής γλώσσας σέ μηχανική.
INTERRUPT: Σῆμα πού θά ἀναγκάσει τόν ὑπολογιστή νά μεταβεῖ ἀπό μιά λειτουργία σέ ἄλλη.
1/0: Εἴσοδος / Ἔξοδος.
K: Συντομογραφία τοῦ 2^{10} = 1028.
LANGUAGE: Μέσο ἐπικοινωνίας μ' ἕνα ὑπολογιστή.
LATCH: Κρατάει μιά κατάσταση εἰσόδου μέχρι νά διαγραφεῖ αὐτή.
LOOPING: Τεχνική προγραμματισμοῦ, ὅπου ἕνα μέρος τοῦ προγράμματος (loop) γίνεται πολλές φορές.
MACHINE LANG: Ἡ μόνη γλώσσα πού καταλαβαίνει ἕνας ὑπολογιστής χωρίς μετάφραση.
MASK: Ἐπιλογή ἑνός bit ἤ bits ἀπό «μηχανική» λέξη.
MEMORY: Μνήμη.
MEMORY MAP: Σχεδιάγραμμα τῆς κατάστασης τῆς μνήμης.
MEMORY MAPPED 1/0: Τεχνική κατά τήν ὁποία 1/0 χρησιμοποιεῖται σάν νά ἐπρόκειτο γιά μνήμη.
MICRO CYCLE: Ἡ μικρότερη διαδικασία.
MICRO PROCESSOR : Ὑπολογιστής ἀποτελούμενος ἀπό πολλά ὁλοκληρωμένα κυκλώματα, πολλές φορές τοποθετημένα σ' ἕνα τσίπ.
MICRO PROGRAM: Πρόγραμμα μέσα στόν ὑπολογιστή πού ρυθμίζει τή βασική διαδικασία.
MNEMONIC: Λέξη ἤ φράση πού ἀντιπροσωπεύει κάτι ἄλλο καί εἶναι πιό εὔκολο γιά ἀποστήθιση.
MODEM: Διαμορφωτής / Ἀποδιαμορφωτής, χρησιμεύει νά στέλνει ἤ νά παίρνει πληροφορίες.
OCTAL: Ὀκταδικό σύστημα. Χρησιμοποιοῦνται οἱ ἀριθμοί 0-8.
OP—CODE: Σχέδιο πού ἐξειδικεύει τή μηχανική λειτουργία.
OPERAND: Πληροφορίες πού χρησιμοποιοῦνται ἀπό τίς μηχανικές λειτουργίες.



PARALLEL: Μεταφορά δύο ἤ περισσότερων bits συγχρόνως.
PARITY: Στή μονή parity τό ἄθροισμα τῶν 1, εἶναι μονή, στή ζυγή, ἀνάποδα.
PERIPHERAL: Ἐργαλεῖα πού συνδέονται στήν εἴσοδο ἤ ἔξοδο τοῦ συστήματος.
PORT: Ἄκρο μέ τό ὁποῖο γίνεται ἡ ἐπικοινωνία.
PROGRAMS: Ὁδηγίες γιά μιά ὁρισμένη ἐργασία.
PROGRAM COUNTER: Ἀποθήκευση τῆς ἑπόμενης πρός ἐκτέλεση διαδικασίας.
PROM: Προγραμματιζόμενη ἀνάγνωση τῆς μνήμης.
RAM: Ἀνάγνωση - Γραφή Μνήμης
REGISTER: Ἀποθήκευση μιᾶς μόνο λέξης.
RELATIVE ADRESSING: Τό operand σχηματίζεται συνδυάζοντας τό παρόν πρόγραμμα μέ ἕνα ἄλλο τμῆμα τῆς διαδικασίας.
ROM: Τό μέρος τῆς μνήμης ὅπου οἱ πληροφορίες εἶναι τοποθετημένες ἐκ κατασκευῆς καί δέν ἀλλάζουν.
SCRATCH PAD: Μνήμη σύντομης διάρκειας.
SERIAL: Μεταφορά πληροφοριῶν ἑνός bit κάθε φορά.
SOFTWARE: Προγράμματα ἀποθηκευμένα σ᾽ ὁποιοδήποτε μέσο.
SOURCE CODE: Τό σύνολο τῶν προτάσεων πού φτιάχνουν ἕνα πρόγραμμα.
STACK: Χῶρος ἀποθήκευσης.
SUB ROUTINE: Μικροπρογράμματα πού ἐπαναλαμβάνονται κατά τήν ἐκτέλεση τοῦ κύριου προγράμματος.
TRI STATE: Λογικές συσκευές τῶν ὁποίων οἱ ἔξοδοι γίνονται ἄχρηστοι, ὅταν τοποθετηθοῦν σέ ψηλῆς ἀγωγιμότητας χῶρο.
TTY: Τηλέτυπος.
TWO'S COMPLEMENT ARITHMETIC: Σύστημα ἐκτέλεσης ἀριθμητικῶν πράξεων μέ δυαδικούς ἀριθμούς.
VDU: Ὀθόνη.
VECTOR: Μεταφορά ἀπό ἕνα τμῆμα τῆς μνήμης σέ ἄλλο.
WORD: Παράλληλη σύνδεση δυαδικῶν ψηφίων, ὄχι περισσότερο ἀπό ἕνα byte.

WICON

ΤΕΛΕΙΑ ΥΛΙΚΑ ΓΙΑ CROSSOVERS



* Ἐπί τέλους, φτάσανε καί στήν Ἑλλάδα τά παγκοσμίου φήμης ὑλικά γιά crossovers τῆς WICON (ἐργοστάσιο εἰδικῆς κατασκευῆς πυρήνων καί πυκνωτῶν γιά κροσσόβερς καί προμηθευτές ὑλικῶν τῶν φημισμένων ἐργοστασίων Peerless, Seas, B&O, SCANIA, JAMO, κλπ).

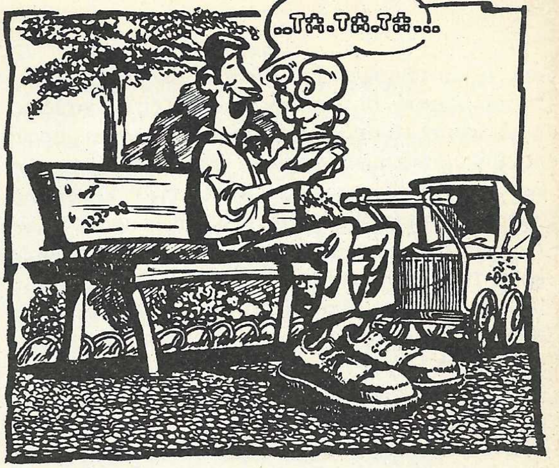
* Εἰδικές προσφορές γιά βιοτέχνες, βιομήχανους ἠχείων, κλπ.

* Ἄν εἶσαι καθηγητής ἤ σπουδαστής ἔλα νά πάρεις τήν εἰδική θεωρία πυρήνων καί πυκνωτῶν γιά κροσσόβερς ΔΩΡΕΑΝ.

* Προσοχή! Τονίζουμε ὅτι οἱ πυρῆνες καί πυκνωτές WICON δέν εἶναι ἁπλῶς πυρῆνες φερίτου ἤ πυκνωτές χωρίς πολικότητα, ἀλλά εἰδικῆς κατασκευῆς πυρῆνες ἀπό εἰδικά κράματα καί πυκνωτές soft foil (διάβασε σχετική θεωρία) μέ ἐλάχιστη ἐσωτερική ἀντίσταση.

ΑΝΤΙΠΡΟΣΩΠΟΙ - ΕΙΣΑΓΩΓΕΙΣ
LYRIC HI-FI
Β. ΘΕΟΔΩΡΟΠΟΥΛΟΣ
ΛΕΩΦ. ΑΛΕΞΑΝΔΡΑΣ
& ΑΣ. ΦΩΤΗΛΑ 32
ΤΗΛ. 8833540 - 8233007 - 8217201

BASIC
η απλη γλωσσα των υπολογιστων



Γράφει ὁ —— Μ. ΟΙΚΟΝΟΜΑΚΗΣ

ΜΕΡΟΣ 2

Ἀριθμοί, ἁπλή ἀριθμητική καί μεταβλητές, εἶναι τά κύρια στοιχεῖα ἑνός προγράμματος BASIC, μέ τά ὁποῖα, θά ἀσχοληθοῦμε στή συνέχεια. Ἐπίσης θά ἀναφερθοῦμε στή μνήμη, ἐντολές, εὕρεση λαθῶν, σύνταξη, κατανομή χώρου, συγκρίσεις καί στίς προτάσεις LET (ἄφησε) καί PRINT (τύπωσε). Ἀλλά ἄς ἀφήσουμε τόν «προγραμματισμό» καί ἄς ἀρχίσουμε τήν ἀνάλυση.

Στό πρῶτο μέρος αὐτῆς τῆς σειρᾶς, εἴχαμε μπεῖ στίς βασικές ἔννοιες καί τά κύρια χαρακτηριστικά τῆς γλώσσας BASIC. Εἶχε τονιστεῖ ἡ ἀνάγκη χρησιμοποίησης "flow charts" (σχεδίων ροῆς). Ἕνα ἁπλό παράδειγμα, ἔδειχνε τή χρήση τῶν ἀριθμημένων γραμμῶν προγράμματος, μπῆκαν δηλ. οἱ προτάσεις "END" καί "PRINT", καί ἐξηγήθηκε ἡ ἐντολή "RUN".

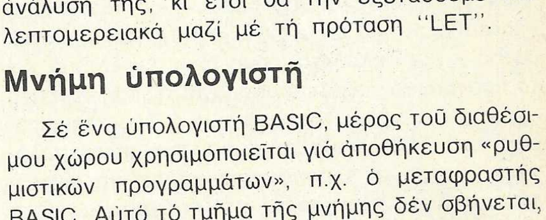
Τό ἑπόμενο βῆμα εἶναι νά μάθουμε τί πρέπει νά γραφτεῖ στίς γραμμές τοῦ προγράμματος: τί ἀριθμούς, πράξεις, μεταβλητές, κλπ. καταλαβαίνει ὁ μεταφραστής BASIC. Ἐπί πλέον, δέν εἶναι δυνατό νά γραφτοῦν καλά προγράμματα ἄν δέν εἶναι, ἔστω καί ἐν μέρει, γνωστές οἱ δυνατότητες τῆς μνήμης τοῦ ὑπολογιστῆ. Μ' αὐτά τά βασικά θά ἀσχοληθοῦμε ἐδῶ.

Ἀφοῦ τά προγράμματα, ἀναπόφευκτα συνδέονται μέ λάθη, ἡ εὕρεση λαθῶν πρέπει ἐπίσης νά ἀναφερθεῖ. Συγχρόνως μιά ἀναφορά σωστῆς χωρικῆς χρησιμοποίησης καί σύνταξης θά βοηθήσει νά εἰσάγονται τά προγράμματα σωστά ἤ νά τά διορθώσει ἀργότερα. Παρ' ὅλο πού ἡ πρόταση "PRINT" ἔχει συζητηθεῖ, δέν ἔχει γίνει τέλεια ἀνάλυσή της, κι ἔτσι θά τήν ἐξετάσουμε πιό λεπτομερειακά μαζί μέ τή πρόταση "LET".

Μνήμη ὑπολογιστῆ

Σέ ἕνα ὑπολογιστή BASIC, μέρος τοῦ διαθέσιμου χώρου χρησιμοποιεῖται γιά ἀποθήκευση «ρυθμιστικῶν προγραμμάτων», π.χ. ὁ μεταφραστής BASIC. Αὐτό τό τμῆμα τῆς μνήμης δέν σβήνεται, χρησιμοποιεῖ ROMS. Ἀποθηκεύοντας πληροφορίες σ' ἕνα ROM, εἶναι ἡ διαδικασία πού γίνεται μόνο μιά φορά, καί γίνεται συνήθως ἀπό τόν κατασκευαστή. Μόλις εἰσαχθεῖ, ἡ πληροφορία διαβάζεται ὅσο συχνά κι ἄν ἀπαιτεῖται, ἀλλά δέν μπορεῖ νά ἀλλοιωθεῖ ἤ νά σβηστεῖ.

Τό ὑπόλοιπο μέρος τῆς μνήμης ἀποτελεῖται κανονικά ἀπό RAMS. Αὐτές προσφέρουν τή δυνατότητα ἀποθήκευσης, ἀνάγνωσης, ἀλλοίω-



Σχ. 1: Τά διάφορα τμήματα μιᾶς μνήμης ὑπολο-

σης ἤ καί σβησίματος πληροφοριῶν, ὁπότε ἀπαιτεῖται. Ὅμως, οἱ πληροφορίες αὐτές θά χαθοῦν, ἄν διακοπεῖ τό ρεῦμα. Ἔτσι μιά πιό μόνιμη μορφή ἀποθήκευσης εἶναι χρήσιμη ὅπως οἱ μαγνητικές κασσέτες ἤ "floppy disc". Αὐτές εἶναι ἐξαιρετικά χρήσιμες γιά ἀποθήκευση ὁλόκληρων προγραμμάτων, ἀλλά ὄχι καί γιά ἀποθήκευση πληροφοριῶν κατά τή διεξαγωγή προγραμμάτων.

Κατά τό γράψιμο καί τή διεξαγωγή προγραμμάτων, τά RAMS εἶναι βασικῆς σπουδαιότητας. Μόνο ἕνα τμῆμα τῆς μνήμης χρησιμοποιεῖται γιά ἀποθήκευση τοῦ προγράμματος, πού λέγεται μνήμη προγράμματος. Ἡ μνήμη αὐτή σβήνεται, ἐνῶ πληροφορίες της μπορεῖ νά τοποθετηθοῦν σέ ἄλλα τμήματα.

Ὅταν χρησιμοποιεῖται NIBL ἡ ὁλική περιοχή RAM διαιρεῖται σέ «σελίδες» (pages). Προγράμματα μποροῦν νά ἀποθηκευτοῦν σέ μία ἤ περισσότερες ἀπό τίς σελίδες 1 ἕως 7.

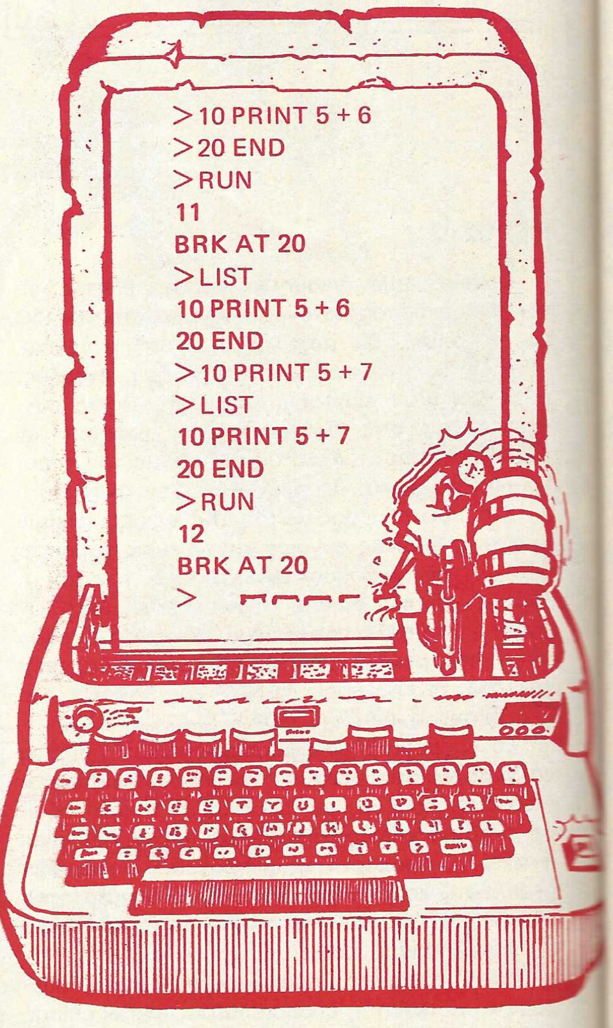
Ρυθμιστές ἐντολῶν

Ὅταν ρυθμιστές ἐντολῶν εἰσάγονται στόν ὑπολογιστή, ἐκτελοῦνται ἀμέσως - ἀντίθετα ἀπό «προτάσεις» πού εἰσάγονται σάν τμῆμα προγράμματος καί ἐκτελοῦνται μόνο ὅταν ἐκτελεῖται καί τό πρόγραμμα. Ὅταν ὁ ὑπολογιστής τυπώσει ἕνα "prompt" σύμβολο, περιμένει κι ἄλλες πληροφορίες ἀπ᾽ τόν ἐκτελεστή (μέσω τοῦ πληκτρολόγιου). Αὐτές οἱ πληροφορίες μπορεῖ νά εἶναι ἡ διαταγή ἤ μιά νέα γραμμή στό πρόγραμμα. Μετά τή εἴσοδο, ὁ ἐκτελεστής χρησιμοποιεῖ τό πλῆκτρο CR (ὁπότε ἐκτελεῖται ἡ διαταγή). Βέβαια αὐτά, λίγο πολύ ἦταν γνωστά ἀπό τό 1ο μέρος. Τώρα θά δοῦμε ποιές διαταγές ὁ (BASIC) ὑπολογιστής ἀναγνωρίζει:

α) RUN: Ὅπως ἐξηγήθηκε στό 1ο μέρος, μόλις εἰσαχθεῖ ἕνα τέλειο πρόγραμμα, δίνεται ἡ διαταγή "RUN". Ὁ ὑπολογιστής θά ἀρχίσει τήν ἐκτέλεση ἀπό τήν 1η γραμμή (ἤ αὐτή πού ἔχει τό μικρότερο ἀριθμό).

Σέ μερικές περιπτώσεις (NIBL καί στό Motorola M6800 BASIC, π.χ.), γίνεται λίγη «δουλειά» ἀκόμα πρίν ὁ ὑπολογιστής ἀρχίσει τήν ἐκτέλεση. Ἀφοῦ πάρει τή διαταγή RUN, μηδενίζει ὅλες τίς «μεταβλητές» καί ἐπαναφέρει ὅλες τίς παραμέτρους τοῦ προγράμματος (αὐτά θά ἐξεταστοῦν πάρα κάτω).

β) LIST: Αὐτή ἡ ἐντολή εἶναι παρόμοια μέ τήν "PRINT" ἤ πιό σωστά, μέ τή μή ὑπάρχουσα ἐντολή "PRINT PROGRAM". Ὅταν ὁ ὑπολογιστής πάρει αὐτή τήν ἐντολή, ἀπαντᾶ τυπώνοντας ὅλο τό πρόγραμμα, ὅπως ἔχει καταγραφεῖ στή μνήμη προγράμματος. Ἄς φέρουμε ἕνα πρόγραμμα γιά παράδειγμα, (τό πρῶτο πρόγραμμα τοῦ πρώτου μέρους). Ὑποθέτουμε ὅτι ἀφοῦ δώσαμε τήν ἐντολή RUN, ἀνακαλύψαμε ἕνα λάθος: στήν πραγματικότητα θέλαμε νά κάνουμε τήν πρόσθεση 5+7, καί ἡ ἀπάντηση θά ἔπρεπε νά εἶναι 12. Μέ τή διαταγή "LIST", ἀνακαλύπτουμε τό λάθος. Διορθώνουμε τό λάθος τῆς σειρᾶς 10, καί γιά νά βεβαιωθοῦμε, ἐπαναλαμβάνουμε τή διαταγή LIST. Ἡ ὁλική καταγραφή εἶναι ὡς ἐξῆς:



Μερικές γλῶσσες "BASIC" ἔχουν ἐπιπρόσθετες μεταβολές τῆς διαταγῆς LIST. Αὐτές, ὅμως, διαφέρουν ἀπό μιά διάλεκτο σέ ἄλλη. Στό NIBL,

π.χ., "LIST n" (ὅπου n εἶναι ἀριθμός γραμμῆς) σημαίνει: γράψε τό πρόγραμμα ἀπό τή γραμμή n καί πέρα-ἀκόμα κι ἄν ἡ ἴδια γραμμή n δέν χρησιμοποιεῖται. Π.χ.:

```
>10 PRINT 1 + 8
>20 PRINT 1 + 9
>30 PRINT 1 + 10
>40 END
>LIST 25
30 PRINT 1 + 10
40 END
>
```

Στή Motorola BASIC διάλεκτο γιά τό 6800, ὡστόσο, ἡ ἴδια ἐντολή ἔχει διαφορετική ἑρμηνεία: LIST 30, π.χ., ἁπλῶς δείχνει τό περιεχόμενο τῆς γραμμῆς 30. Μερικές διάλεκτοι ἀναγνωρίζουν μιά μεταβολή πού εἶναι ἄγνωστη στό NIBL (ἤ στό DCE Tiny Basic γιά τό 8080): "LIST n, m", ὅπου n, m εἶναι ἀριθμοί γραμμῶν. Σ᾽ αὐτή τήν περίπτωση, ἡ τύπωση τοῦ προγράμματος ἀρχίζει στή γραμμή n καί τελειώνει στή γραμμή m.

γ) PAGE: Ὅπως ἀναφέρθηκε, ἡ μνήμη ὑπολογιστοῦ NIBL ὑποδιαιρεῖται σέ "pages". Ὅταν πρωτοανάψει ἕνας ὑπολογιστής NIBL, αὐτόματα γυρίζει στή «σελίδα» 2 καί ἀρχίζει νά ἐκτελεῖ τό πρόγραμμα πού εἶναι ἀποθηκευμένο ἐκεῖ. Βέβαια αὐτό προϋποθέτει ὅτι οἱ πληροφορίες τῆς «σελίδας» 2 εἶναι ἀποθηκευμένες σέ ROMs, διότι σέ RAMs θά σβηνόταν τό πρόγραμμα μόλις σβηνόταν καί ὁ ὑπολογιστής.

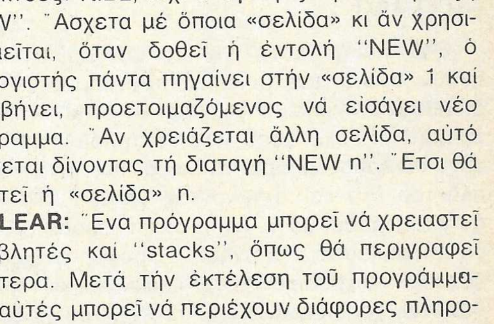
Ἐάν ἡ «σελίδα» 2 εἶναι κενή, αὐτόματα γυρίζει στή «σελίδα» 1 καί τυπώνει ἕνα "prompt" σύμβολο. Ἐάν τυπωθεῖ τώρα ἕνα πρόγραμμα, θά τυπωθεῖ στή «σελίδα» 1. Ὡστόσο, αὐτό μπορεῖ νά εἶναι ἀνεπιθύμητο (π.χ. ἡ «σελίδα» 1 μπορεῖ νά χρειάζεται γιά κάποιο ἄλλο πρόγραμμα). Σ᾽ αὐτή τήν περίπτωση δίνεται ἡ ἐντολή "PAGE=n". Αὐτό ἀναγκάζει τόν ὑπολογιστή νά γυρίσει στή σελίδα n (n=1...7) γιά νά ἀποθηκευτεῖ τό πρόγραμμα ἐκεῖ.

Γενικά, μποροῦμε νά πηδήξουμε ἀπό μιά σελίδα στήν ἀρχή μιᾶς ἄλλης, δίνοντας τήν ἐντολή "PAGE=n". Μπορεῖ νά χρησιμοποιηθεῖ καί μιά παραλλαγή: "PAGE=PAGE+n" ἤ "PAGE=PAGE-n". Αὐτό ἐξηγεῖται καλλίτερα μέ παράδειγμα. Ἄς ὑποθέσουμε ὅτι ὁ ὑπολογιστής βρίσκεται στή σελίδα 3. "PAGE=PAGE-1", μεταφράζεται σάν PAGE=3-1, δηλαδή PAGE=2. Ἔτσι ὁ ὑπολογιστής θά γυρίσει στή «σελίδα» 2. Ἡ νέα «σελίδα» πρέπει νά εἶναι μεταξύ 1–7, διότι δέν ὑπάρχουν ἄλλες «σελίδες».

δ) SCRATCH, DELETE, PURGE, NEW: Διαφορετικοί διάλεκτοι, διαφορετικές λέξεις, ἀλλά ἡ ἴδια ἐντολή.

Τό SCRATCH (σύντομα SCR), ἀναγκάζει τόν ὑπολογιστή νά σβήσει τό ὑπάρχον πρόγραμμα καί τήν ὀθόνη. Μερικές διάλεκτοι BASIC ἀναγνωρίζουν τήν ἐντολή "SCRATCH ALL". Σ᾽ αὐτή τήν περίπτωση ὁλόκληρη ἡ μνήμη RAMs σβήνεται. "DELETE", "PURGE", καί "NEW", εἶναι ἄλλες λέξεις γιά τήν ἴδια ἐντολή σέ διαφορετικές διαλέκτους. NIBL, π.χ. ἀναγνωρίζει μόνο τή λέξη "NEW". Ἄσχετα μέ ὅποια «σελίδα» κι ἄν χρησιμοποιεῖται, ὅταν δοθεῖ ἡ ἐντολή "NEW", ὁ ὑπολογιστής πάντα πηγαίνει στήν «σελίδα» 1 καί τή σβήνει, προετοιμαζόμενος νά εἰσάγει νέο πρόγραμμα. Ἄν χρειάζεται ἄλλη σελίδα, αὐτό δείχνεται δίνοντας τή διαταγή "NEW n". Ἔτσι θά σβηστεῖ ἡ «σελίδα» n.

ε) CLEAR: Ἕνα πρόγραμμα μπορεῖ νά χρειαστεῖ μεταβλητές καί "stacks", ὅπως θά περιγραφεῖ ἀργότερα. Μετά τήν ἐκτέλεση τοῦ προγράμματος, αὐτές μπορεῖ νά περιέχουν διάφορες πληροφορίες πού δέν χρειάζονται ἄλλο (ἐνδιάμεσα ἀποτελέσματα, κ.ἄ.). Πρίν ξαναεκτελεστεῖ τό πρόγραμμα, αὐτά μπορεῖ νά σβηστοῦν μέ τή διαταγή "CLEAR".



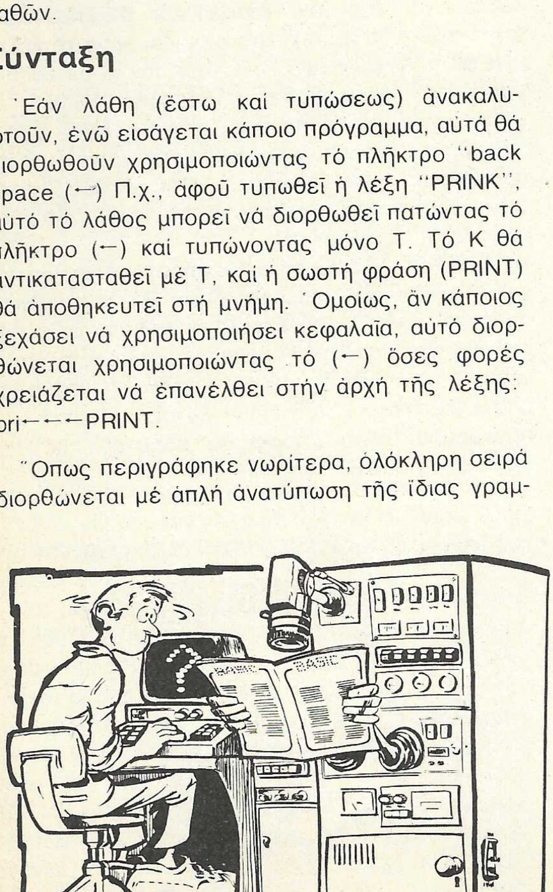
Λάθος σύνταξης

Όταν πρόκειται νά γραφεῖ ἕνα πρόγραμμα, πολλά μπορεῖ νά μᾶς ξεφύγουν. Ἀκόμα κι ὅταν τό ἴδιο τό πρόγραμμα εἶναι τέλειο, ὑπάρχει πάντα ἡ πιθανότητα νά τυπωθεῖ κάποιο λάθος κατά τήν εἰσαγωγή του στόν ὑπολογιστή. Τό παραμικρό λάθος αὐτοῦ τοῦ εἴδους - τυπώνοντας RAN ἀντί γιά RUN ἤ PRANT ἀντί γιά PRINT - θά κάνει ἄχρηστη τήν ἐντολή ἤ πρόταση γιά τόν ὑπολογιστή. Εὐτυχῶς τά λάθη τυπώσεως ἀναγνωρίζονται συνήθως καί θά προειδοποιηθεῖ ὁ ἐκτελεστής. Ἕνα παράδειγμα προειδοποιητῆ εἶναι ἡ φράση "SYNTAX ERROR" (Λάθος Σύνταξης), παρ' ὅλο πού οἱ ὑπολογιστές συνήθως χρησιμοποιοῦν ποιό ἀγενή ὁρολογία... SYNTAX ERROR (μερικές φορές, ὅπως στό NIBL, σύντομα σάν SNTX ERROR) προειδοποιεῖ γιά «κακή φρασιολογία», δηλαδή ἡ φράση πού τυπώθηκε δέν ὑπάρχει στή δεδομένη BASIC διάλεκτο. Ὑπάρχουν κι ἄλλα ἐνδεικτικά λαθῶν.

Σύνταξη

Ἐάν λάθη (ἔστω καί τυπώσεως) ἀνακαλυφτοῦν, ἐνῶ εἰσάγεται κάποιο πρόγραμμα, αὐτά θά διορθωθοῦν χρησιμοποιώντας τό πλῆκτρο "back space (←) Π.χ., ἀφοῦ τυπωθεῖ ἡ λέξη "PRINK", αὐτό τό λάθος μπορεῖ νά διορθωθεῖ πατώντας τό πλῆκτρο (←) καί τυπώνοντας μόνο Τ. Τό Κ θά ἀντικατασταθεῖ μέ Τ, καί ἡ σωστή φράση (PRINT) θά ἀποθηκευτεῖ στή μνήμη. Ὁμοίως, ἄν κάποιος ξεχάσει νά χρησιμοποιήσει κεφαλαῖα, αὐτό διορθώνεται χρησιμοποιώντας τό (←) ὅσες φορές χρειάζεται νά ἐπανέλθει στήν ἀρχή τῆς λέξης: pri←←←PRINT.

Ὅπως περιγράφηκε νωρίτερα, ὁλόκληρη σειρά διορθώνεται μέ ἁπλή ἀνατύπωση τῆς ἴδιας γραμμῆς σωστά. Ἐξ ἄλλου ὁλόκληρη γραμμή σβήνεται πιέζοντας τό πλῆκτρο CR.



Οἱ περισσότερες διάλεκτοι BASIC ἔχουν κι ἄλλες εὐκολίες σύνταξης, ἀλλά οἱ ἀκριβεῖς λεπτομέρειες συνήθως ὑπαγορεύονται ἀπό τόν τύπο τοῦ πληκτρολόγιου. NIBL, π.χ. ἔχει τήν ἐντολή "CONTROL/U" - δηλ. τά πλῆκτρα Control καί U πιέζονται συγχρόνως. Σ' αὐτή τήν περίπτωση ἡ τρέχουσα γραμμή (ἡ γραμμή πού τυπώνεται ἐκείνη τή στιγμή), σβήνεται ἀπό τήν ὀθόνη - ἀλλά ὄχι ἀπό τή μνήμη προγράμματος. Αὐτή ἡ εὐκολία εἶναι πολλή χρήσιμη, ἄν τυπωθεῖ λάθος γραμμή: τό λάθος διορθώνεται χωρίς νά χαθοῦν ὁδηγίες ἤδη ἀποθηκευμένες στήν ἀριθμημένη γραμμή στή μνήμη.

Κατανομή χώρου

Ὅταν τυπώνεται ἕνα κύκλωμα, εἶναι χρήσιμο νά προστεθοῦν κενά ἐδῶ κι ἐκεῖ - π.χ. ἀνάμεσα σέ προτάσεις. Παρ' ὅλο πού ὁ ὑπολογιστής δέν ἔχει τήν παραμικρή ἰδέα περί τίνος πρόκειται, καί κανονικά θά τά ἀγνοήσει, τά ἀποθηκεύει στή μνήμη καί τά παρουσιάζει μέ τήν ἐντολή "LIST". Ὁ κύριος λόγος πρόσθεσης κενῶν, εἶναι γιά νά κάνει τό πρόγραμμα εὔχρηστο γιά τόν ἐκτελεστή. Σέ μερικές ὅμως περιπτώσεις ἀπαγορεύεται ἡ χρήση τους:

- Μέσα σέ λέξεις πού εἶναι μέρος πρότασης ἤ ἐντολῆς. π.χ. P R INT εἶναι λάθος, πρέπει νά γραφτεῖ PRINT. Ὡστόσο PRINT5 καί PRINT 5, ἐπιτρέπονται καί τά δύο.
- Ἀνάμεσα σέ ἀριθμούς (περιλαμβανομένων καί ἀριθμῶν σειρῶν). Ἡ γραμμή προγράμματος '150 PRINT 2500' εἶναι σωστή, ἐνῶ ''1 50 PRINT 2500'' καί ''150 PRINT 2 500'' εἶναι λάθος. Πρέπει ὡστόσο νά σημειωθεῖ ὅτι '150 PRINT ''250''' εἶναι σωστό. Ἄλλες ἀπαγορευμένες περιπτώσεις θά ἀναφερθοῦν ὅταν τίς συναντήσουμε. Ἐξ ἄλλου, ὁρισμένες διάλεκτοι τῆς BASIC ἀπαιτοῦν χρήση κενῶν, π.χ. πρίν καί ἤ μετά προτάσεις ἤ ἐντολές. Αὐτό δέν θά εἶναι κανονικά πρόβλημα. Ὁ χῶρος σέ τέτοιες περιπτώσεις ἀφήνεται ἀσυναίσθητα. Π.χ. (χρησιμοποιώντας μερικές προτάσεις πού θά περιγραφοῦν στό 3ο μέρος), θά ἦταν σχεδόν παράδοξο τό ἑξῆς.

```
10IFA=BLETA=B–C.
```

Ἐνῶ προσθέτοντας λίγα κενά, μετατρέπει τήν πρόταση σέ ἁπλά Ἀγγλικά:

```
10 IF A=B LET A=B-C.
```

΄Αν ἐπιτρέπεται ἕνα κενό, ἐπιτρέπονται καί περισσότερα, π.χ.:

```
10 IF A=B  LET A=B-C.
```

΄Αριθμοί

Στή BASIC, οἱ ἀριθμοί μπορεῖ νά γραφτοῦν μέ τό συνηθισμένο τρόπο. Ωστόσο, πρέπει νά τονιστοῦν σημεῖα.

α) ΄Ακέραιοι: Αριθμοί πού δέν περιέχουν κλάσματα ἤ πού δέν εἶναι δεκαδικοί, λέγονται ἀκέραιοι. Αὐτοί μπορεῖ νά εἶναι θετικοί ἤ ἀρνητικοί. ΄Εάν εἶναι θετικοί μπορεῖ νά συνοδεύονται μέ τό πρόσημο +, ἐάν εἶναι ἀρνητικοί πρέπει νά συνοδεύονται μέ τό πρόσημο –. ΄Οπως ἀναφέρθηκε δέν ἐπιτρέπεται νά ἀφήνουμε χῶρο ἀνάμεσα στούς ἀριθμούς. Π.χ.

Σωστά: 3, +3, +123456789, -3, -567

Λάθος: 123 456

β) Κλάσματα: Πιό σωστά, δεκαδικά κλάσματα: ἀριθμοί πού περιέχουν ὑποδιαστολή. ΄Οπως, πρίν, '+' ἤ '-1', μπορεῖ ἤ πρέπει ἀντίστοιχα νά χρησιμοποιοῦνται. Σέ μερικές διαλέκτους BASIC πρίν τήν ὑποδιαστολή δέν πρέπει νά τοποθετεῖται 0, π.χ. .38 εἶναι ἐντάξει, ἐνῶ 0.38 ὄχι. Πάλι μερικά παραδείγματα.

Σωστά: 2.2, +1.23, -55.5, -44.

Λάθος: 2 1/2, -3/4.

Στή NIBL χρησιμοποιοῦνται μόνο ἀκέραιοι. Κάθε προσπάθεια χρησιμοποίησης ὑποδιαστολῆς φέρνει στήν ὀθόνη τή λέξη 'CHAR ERROR' (character error = λάθος συμβόλου).

γ) Περιοχή ἀριθμῶν: Στίς περισσότερες διαλέκτους BASIC ὁ μέγιστος ἀριθμός ψηφίων εἶναι ἐννέα - χωρίς τά '+' ἤ '–' καί / ἤ τήν ὑποδιαστολή. Π.χ.: -123456.789. ΄Ετσι ὁ μεγαλύτερος ἀριθμός πού μπορεῖ νά γραφτεῖ εἶναι ὁ 999999999 καί ὁ μικρότερος (θετικός) .000000001. Στή NIBL ὁ ἀριθμός τῶν ψηφίων εἶναι πιό περιορισμένος: μόνο ἀκέραιοι μεταξύ -32767 καί +32767 ἐπιτρέπονται. Αὐτά τά ὅρια ἀνταποκρίνονται στό μεγαλύτερο ἀριθμό πού μπορεῖ νά γραφτεῖ σ' ἕνα δυαδικό σύστημα 16-bit. ΄Εάν εἰσαχθεῖ μεγαλύτερος ἀριθμός στήν ὀθόνη, καταγράφεται ἡ λέξη 'VALU ERROR' (value error = λάθος τιμῆς). Π.χ.:

```
> PRINT 44253
VALU ERROR
>
```

δ) ΄Αριθμητική ἐπέκταση: Σέ μερικές περιπτώσεις, η περιοχή τῶν ἀριθμῶν μπορεῖ νά ἀποδειχτεῖ πολύ περιορισμένη. Γι' αὐτό πολλές διάλεκτοι BASIC ἔχουν καί μιά ἐπεκτατική εὐκολία: 'scientific notation', ἐπίσης γνωστή σάν "E nubers" (ἀριθμοί E). Βασικά, ἀποτελεῖται ἀπό ἕνα ἀριθμό, ἀκολουθούμενο ἀπό τό γράμμα E καί μετά ἀπό δύο ἀκόμα ψηφία. Αὐτά τά δύο τελευταῖα ψηφία καθορίζουν πόσες, θέσεις δεξιά μετατοπίζεται ἡ ὑποδιαστολή (θετικός ἀριθμός μετά τό E) ἤ πρός τά ἀριστερά (ἀρνητικός ἀριθμός μετά τό E). Μέ ἄλλες (μαθηματικές) λέξεις: ὁ ἀριθμός πολλαπλασιάζεται μέ μία δύναμη τοῦ δέκα, ὅπως καθορίζεται ἀπό τό διψήφιο ἀριθμό. Μερικά παραδείγματα διασαφηνίζουν τή λειτουργία:

4.35E5 = 435000

1234.5 E-3=1.2345

Αὐτά δέν γίνονται στή NIBL

ε) ΄Αριθμητική ἀκρίβεια: Η ἀκρίβεια πού διακρίνει τόν ὑπολογιστή, ὅσο ἀφορᾶ ἀριθμούς (ὅταν τά ἀποθηκεύει στή μνήμη ἤ κάνει ὑπολογισμούς), ἐξαρτᾶται τόσο ἀπό τό μεταφραστή, ὅσο κι ἀπό τόν ἴδιο τόν ὑπολογιστή. ΄Η ἀκρίβεια κανονικά εἶναι μεταξύ 5 καί 7 ψηφίων. Μεγαλύτερος ἀριθμός ψηφίων «διώχνεται», δηλ. ὁ ἀριθμός 123456789 μπορεῖ νά ἀπλοποιηθεῖ σάν 123450000, 123456000, ἤ 123456700.

΄Αριθμητικές πράξεις: Στή BASIC ἐπιτρέπονται 5 ἀριθμητικές πράξεις: +, -, *, /, καί↑. ΄Η σημασία τους καί μερικά παραδείγματα φαίνονται στόν πάρα κάτω πίνακα

Πίνακας 1

Λειτουργία	πράξη	αποτέλεσμα	ἐξήγηση
+	3 + 5	8	πρόσθεση
–	3 – 5	–2	ἀφαίρεση
*	3 * 5	15	πολ/σμός
/	6 / 2	3	διαίρεση
↑	2 ↑ 3	8	ὕψωση στή νιοστή δύναμη

΄Οταν περιλαμβάνονται πάνω ἀπό μία πράξη, γίνονται μέ τή σωστή σειρά: ὕψωση σέ δύναμη, πολλαπλασιασμός ἤ διαίρεση, καί τέλος πρόσθεση ἤ ἀφαίρεση. Π.χ. 6+4/2 ὑπολογίζεται ὡς ἀκολούθως: 4/2=2 6+2=8. ΄Εάν ἡ πρόσθεση ἤ ἀφαίρεση πρέπει νά γίνει πρώτη, αὐτό δείχνεται βάζοντας τήν ἐν λόγω πράξη σέ παρένθεση. (6+4)/2 ὑπολογίζεται ὡς ἑξῆς: 6+4=10 10/2=5.

Συγκρίσεις: ΄Αριθμητικές συγκρίσεις, ὅπως εἶναι A > B συχνά ἀπαιτοῦνται σέ προγράμματα. Τό ἀποτέλεσμα μιᾶς συγκρίσεως μπορεῖ νά εἶναι εἴτε

ἀληθές (΄1΄) εἴτε ψευδές (΄0΄). Στό πάρα κάτω πίνακα φαίνονται τά διάφορα σύμβολα συγκρίσεως μέ μερικά παραδείγματα.

ΠΙΝΑΚΑ 2

σύμβολο	σημασία	παράδειγμα	ἀποτέλεσμα
=	(A) ἴσον (B)	3 = 4	ψευδές (0)
< > ή > <	(A) ≠ (B)	3 < > 4	ἀληθές (1)
>	(A) > (B)	3 > 4	ψευδές (0)
> = η = >	(A) ≥ (B)	3 > = 4	ψευδές (0)
<	(A) < (B)	3 < 4	ἀληθές (1)
< = ή = <	(A) ≤ (B)	3 < = 4	ἀληθές (1)

Στήν περίπτωση ὅπου φαίνονται δύο σύμβολα (<> ή >< γιά ≠) ή NIBL χρησιμοποιεῖ μόνο τό πρῶτο. Στίς περισσότερες διαλέκτους BASIC χῶρος ἀνάμεσα σέ δύο μέρη ἑνός συμβόλου δέν ἐπιτρέπεται. Π.χ. > = δέν πρέπει νά τυπωθεῖ σάν > =.

Μεταβλητές: Πρόκειται γιά τήν ὀνομασία πού μιά ἀριθμητική ἀξία παίρνει. Ἡ χρήση τους φαίνεται μέ παράδειγμα. Ἄς ὑποθέσουμε ὅτι θέλουμε νά ὑπολογίσουμε τό μέγιστο ρεῦμα ἐξόδου (I) κάποιου κυκλώματος, καί ὅτι αὐτό τό ρεῦμα ἐξαρτᾶται ἀπό τή τάση (U) καί μιά ἀντίσταση (R) ὡς ἑξῆς: I=U/2R. Δέν εἶναι δύσκολο νά γραφτεῖ τό ἀκόλουθο κατάλληλο πρόγραμμα:

```
> 10 PRINT U/2*R
> 20 END.
```

Σ᾽ αὐτό τό πρόγραμμα τά U καί R εἶναι μεταβλητές. Ἀφοῦ δοθοῦν οἱ τιμές τους, τό πρόγραμμα ξεκινάει μέ τήν ἐντολή RUN καί τό σωστό ἀποτέλεσμα θά δοθεῖ. Ὁλόκληρο τό πρόγραμμα μέ μιά μικροβελτίωση στή γραμμή 15 ἔχει ὡς ἑξῆς:

```
> 10 PRINT U/2*R;
> 15 PRINT "AMPERE"
> 20 END
> U = 10
> R = 5
> RUN
1 AMPERE
BRK AT 20
>
```

Τό πλεονέκτημα τῆς χρήσης μεταβλητῶν εἶναι ὅτι γίνονται νέοι ὑπολογισμοί γιά διαφορετικές τιμές τῶν μεταβλητῶν, χωρίς νά πρέπει νά ξαναγραφτεῖ ἕνα ὁλόκληρο κομμάτι τοῦ προγράμματος. Ἡ ἀνωτέρω τύπωση μπορεῖ νά συνεχιστεῖ, ἀπό τή γραμμή μετά 'BRK AT 20', ὡς ἑξῆς:

```
> U = 2000
> R = 2
> RUN
500 AMPERE
BRK AT 20
>
```

Βέβαια τό ἀνωτέρω πρόγραμμα δέν θά ἐκτελεσθεῖ τελείως σέ ὅλους τούς ὑπολογιστές. Ὁ λόγος εἶναι ὅτι, σέ μερικές διαλέκτους BASIC, ἡ ἐντολή RUN ἀναγκάζει ὅλες τίς μεταβλητές νά μηδενιστοῦν πρίν ἀρχίσει τό πρόγραμμα. Αὐτό εἶναι χρήσιμο, στό ὅτι ἐλαχιστοποιεῖ τήν πιθανότητα ἐκτέλεσης τοῦ προγράμματος μέ παλιές τιμές γιά τίς μεταβλητές. Ὡστόσο, σημαίνει ἐπίσης ὅτι οἱ μεταβλητές παίρνουν τιμές μόνο κατά τήν ἐκτέλεση τοῦ προγράμματος - ὄχι προηγουμένως. Αὐτό πετυχαίνεται ἀρκετά εὔκολα προσθέτοντας ὁσεσδήποτε γραμμές προγράμματος ἀπαιτοῦνται πρίν ἀπό τήν πρώτη κανονική γραμμή τοῦ προγράμματος. Στό παραπάνω παράδειγμα, δηλ. καί προσθέτοντας μία διαταγή 'LIST' γιά νά ἐλεγχθεῖ τό πρόγραμμα πρίν χρησιμοποιηθεῖ:

```
> 5 U = 2000
> 6 U = 2
> LIST
5 U = 2000
6 R = 2
10 PRINT U/R*2;
15 PRINT "AMPERE"
20 END
> RUN
500 AMPERE
BRK AT 20
>
```

A	A0	A1	A2	A3	A4	A8	A9
B	B0	B1	B2	B3	B4	B8	B9
C	C0	C1	C2	C3	C4	C8	C9
D	D0	D1	D2	D3	D4	D8	D9
E	E0	E1	E2	E3	E4	E8	E9
F	F0	F1	F2	F3	F4	F8	F9
G	G0	G1	G2	G3	G4	G8	G9
X	X0	X1	X2	X3	X4	X8	X9
Y	Y0	Y1	Y2	Y3	Y4	Y8	Y9
Z	Z0	Z1	Z2	Z3	Z4	Z8	Z9

Σχ. 2: Εἶναι σκόπιμο νά κρατιέται κατάλογος τῶν «πραγματικῶν» ἐννοιῶν πού συμβολίζουν οἱ μεταβλητές.

Σέ μερικές διαλέκτους χρησιμοποιοῦνται ἀριθμοί. Στίς περισσότερες, ὅμως, περιπτώσεις ἕνα γράμμα ἀκολουθούμενο ἀπό ἕνα ἀριθμό ἐπιτρέπεται. Ἔτσι δέν εἶναι ἀσυνήθιστοι οἱ συμβολισμοί A,D,D1,Z9, ἀλλά ὄχι οἱ AZ, G12, κτλ. Ἔτσι μποροῦν νά χρησιμοποιηθοῦν μέχρι 286 διαφορετικές μεταβλητές. Μάλιστα σέ μεγάλα προγράμματα, μπορεῖ ὁ ἐκτελεστής νά ξεχνᾶ τί ἀντιπροσωπεύει ἕνας ὁρισμένος συμβολισμός καί νά ἐκτελεῖται ἕνα πρόγραμμα χωρίς ὁ ἐκτελεστής νά ἀντιληφθεῖ ὅτι τά ἀποτελέσματα εἶναι λάθος.

Ἔτσι εἶναι σκόπιμο νά ὑπάρχει ἕνας κατάλογος, ὅπου νά φαίνεται τό κάθε σύμβολο τί ἀντιπροσωπεύει. Ὁ συστηματικός κατάλογος τοῦ σχ. 2 εἶναι συνήθως ὁ καλλίτερος. Στή NIBL μόνο τά 26 γράμματα τῆς ἀλφαβήτου ἐπιτρέπονται, χωρίς νά εἶναι εὔκολη ἡ σύγχυση ἐδῶ. Ἐκτός ἀπ' αὐτές τίς κοινά λεγόμενες «ἁπλές» μεταβλητές, ὑπάρχει κι ἕνας δεύτερος τύπος πού λέγεται 'string variables' (ἀλυσίδα μεταβλητῶν), ὅπου οἱ μεταβλητές ἀντιπροσωπεύουν μιά ἀλυσίδα χαρακτήρων (ἀριθμῶν καί γραμμάτων). Αὐτή η κατηγορία θά ἐξεταστεῖ ἀργότερα.

LET

Τό LET (ἄφησε) εἶναι μιά πρόταση ἀναθέσεως, ὅπως λέγεται. Χρησιμοποιεῖται γιά νά ἀναθέσει μιά ὁρισμένη τιμή σέ μιά μεταβλητή. Στά προηγούμενα παραδείγματα προγραμμάτων, αὐτό γινόταν π.χ. τυπώνοντας, 'U = 2000'. Παρ' ὅλο πού οἱ περισσότερες διάλεκτοι BASIC, ἀνέχονται αὐτή τήν (κακή) χρήση τοῦ συμβόλου '=', δέν εἶναι ὁ σωστός τρόπος νά δοθεῖ τιμή σέ μία μεταβλητή. Εἶναι πιό σωστό νά χρησιμοποιηθεῖ μιά πρόταση LET. Ὁλόκληρη ἡ ὁδηγία τυπώνεται τότε ὡς ἑξῆς: πρῶτα 'LET', μετά τό 'ὄνομα' τῆς μεταβλητῆς, μετά τό σύμβολο '=' καί τέλος ἡ 'ἔκφραση'. Μερικά παραδείγματα:

```
LET A=15
LET A=B
LET A=3+4
LET A=B+C
```

Ὅπως φαίνεται, μιά μεταβλητή μπορεῖ νά ἐξισωθεῖ μέ περισσότερες μεταβλητές ἤ μέ κάποια ἀριθμητική πράξη, ὅπου συμμετέχουν καί ἄλλες μεταβλητές. Ἐξ ἄλλου, ἡ ἴδια μεταβλητή μπορεῖ νά ἐμφανιστεῖ κι ἀπό τίς δύο μεριές τοῦ '=' συμβόλου. Π.χ. LET A=A+1.

Σ' αὐτή τήν περίπτωση ἡ νέα τιμή τῆς μεταβλητῆς (A) ἐξάγεται ἀπό τήν προηγούμενη. Ἄν A=4, τότε τό νέο A θά εἶναι A=4+1=5. Μερικές BASIC διάλεκτοι, μή περιλαμβανομένης τῆς NIBL, παρέχουν τή δυνατότητα νά δοθοῦν τιμές μερικῶν μεταβλητῶν συγχρόνως. Ἡ ὁδηγία LET A=B=C=15, ἐπιτρέπει καί στίς τρεῖς μεταβλητές νά πάρουν τή τιμή 15. Σέ μερικές διαλέκτους συντομότερα ἐπιτρέπεται ἡ ἔκφραση 'R=5' ἀντί γιά 'LET R=5'. Αὐτό ἐπιτρέπεται καί στή NIBL

Λίγα ἀκόμα γιά τό PRINT: Ἡ πρόταση PRINT μπῆκε στό πρῶτο μέρος. Ἄ θυμηθοῦμε τίς μέχρι τώρα ἀναφερόμενες δυνατότητες σύντομα: PRINT «5+6=». Σ' αὐτή τήν περίπτωση, τό κείμενο μέσα στά εἰσαγωγικά τυπώνεται ἀκριβῶς ὅπως ἔχει:

Σχ. 3: Μέσα σέ μιά πρόταση PRINT, κόμματα χρησιμοποιοῦνται γιά νά χωρίσουν τό τύπωμα σέ περιοχές.

5+6= PRINT 5+6: Πρῶτα γράφεται ἡ πράξη καί μετά τό ἀποτέλεσμα: 11 PRINT. Ἀφοῦ ἡ πρόταση δέν ἀκολουθεῖται ἀπό κείμενο ἤ πράξη, τίποτε δέν γράφεται. Ἔτσι δημιουργεῖται μιᾶς γραμμῆς κενό. Συνήθως μιά πρόταση PRINT ἀκολουθεῖται ἀπό ἕνα CR καί LF. Αὐτές παραλείπονται χρησιμοποιώντας ἄνω τελεία ἀνάμεσα σέ προτάσεις PRINT:

```
10 PRINT "TOM"; " DICK"; " HARRY"
20 PRINT "TOM";
21 PRINT " DICK";
22 PRINT " HARRY"
```

Ἡ γραμμή 10 ἔχει τό ἴδιο νόημα μέ τίς 20,21,22 καί χρησιμοποιεῖται ἔτσι μιά γραμμή.

Μιά ἀκόμα δυνατότητα εἶναι ἡ χρήση κόμματος σέ προτάσεις PRINT PRINT 121,122. Τό ἀποτέλεσμα εἶναι ὅτι διάφορα κείμενα τυπώνονται σέ διάφορες περιοχές. Μιά περιοχή ἔχει 15 χαρακτῆρες. Ἔτσι στό πάρα πάνω παράδειγμα '121' τυπώνεται στήν ἀρχή τῆς γραμμῆς καί '122' στή 16η, 17η καί 18η θέση. Τό μῆκος μιᾶς γραμμῆς εἶναι 72 χαρακτῆρες (ἔτσι περιέχει λίγο λιγότερο ἀπό 5 περιοχές).

Ἡ τελευταία περιοχή ἔχει μόνο 12 χαρακτῆρες. Ἡ χρήση τῶν περιοχῶν εἶναι ἐξαιρετικά μεγάλη ὅταν τυπώνονται πίνακες. Ἡ χρήση τοῦ κόμματος γιά τύπωση σέ περιοχές δέν ἐπιτρέπεται στή NIBL.

(Συνεχίζεται)

Λεξικό ὅρων καί συμβόλων

Assignment statement = ὁδηγία πού δίνει σέ μεταβλητή ἀριθμητική ἀξία π.χ. LET A=1
Control command = ὁδηγία πού πραγματοποιεῖται ἀμέσως κι ὄχι σάν μέρος τοῦ προγράμματος.
current line (program) = Ἡ γραμμή προγράμματος πού τυπώνεται ἐκείνη τή στιγμή
editing = εἰσαγωγή καί διόρθωση προγραμμάτων (μέσω τοῦ πληκτρολόγιου).
error indication = μερικά λάθη προγραμματισμοῦ φαίνονται ἀμέσως, ὁπότε προειδοποιεῖται ὁ ἐκτελεστής.
floppy disc = εὔκαμπτος μαγνητικός δίσκος, πού ἀποθηκεύει μεγάλα ποσά πληροφοριῶν.
integer = ἀκέραιος
listing = μετά ἀπό μιά διαταγή LIST, ὁ ὑπολογιστής τυπώνει τό ὑπάρχον πρόγραμμα. Ἡ τύπωση λέγεται listing
page = ὑπότμημα μνήμης σ' ἕναν ὑπολογιστή NIBL
RAM: τμῆμα τῆς μνήμης πού ἀποθηκεύει πληροφορίες, ὅταν ἀπαιτεῖται.
ROM: τμῆμα τῆς μνήμης ὅπου ἀποθηκεύονται πληροφορίες κατά τή κατασκευή καί οἱ ὁποῖες δέν σβήνονται.
simple numerical varioble = μεταβλητή στήν ὁποία δίνεται κάποια τιμή· αὐτή ἡ τιμή μεταβάλλεται ἐνῶ ἐκτελεῖται τό πρόγραμμα
string variable = μεταβλητή στήν ὁποία δίνεται μιά ὁμάδα χαρακτήρων–μιά λέξη, π.χ.
variable = ὄνομα πού ἀντιπροσωπεύει ἕνα ἀριθμό ἤ σύνολο γραμμάτων, κοινῶς μεταβλητή.

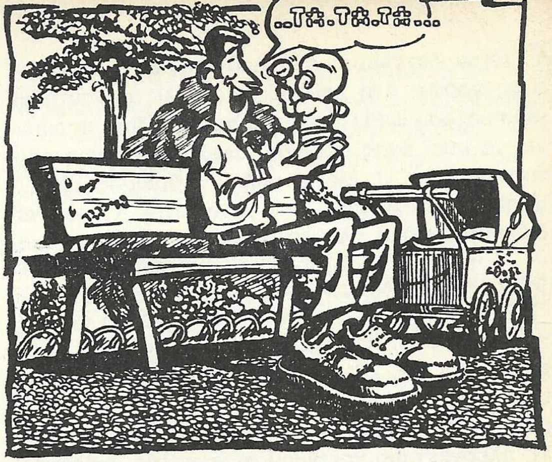
ΒΟΡΕΙΟΗΛΕΚΤΡΟΝΙΚΗ

μεγάλη ποικιλία ἀπό:

- ραδιοτηλέφωνα CB • κεραῖες • καλώδια
- ραδιοκασσετόφωνα αὐτοκινήτου
- ἀκουστικά στέρεο • τηλεφωνικές συσκευές
- φωτιστικά διακοσμητικά

ΠΑΤΡ. ΙΩΑΚΕΙΜ 26 (1ος ὄροφος) ΘΕΣΣΑΛΟΝΙΚΗ
Τ.Θ. 834 - ΤΗΛ. 211147

BASIC
η απλη γλωσσα των υπολογιστων



Γράφει ὁ
Μ. ΟΙΚΟΝΟΜΑΚΗΣ

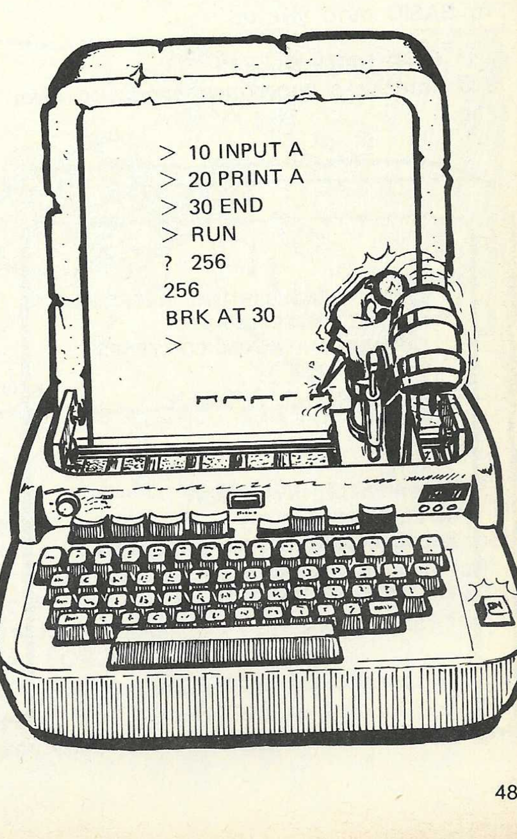
ΜΕΡΟΣ 3

Τά σπουδαιότερα στοιχεῖα προγραμματισμοῦ σέ BASIC ἔχουν ἤδη ἐξεταστεῖ. Σ᾽ αὐτό τό μέρος θά ἐξηγηθοῦν οἱ περισσότερες προτάσεις πού δέν ἔχουν ὥς τώρα ἐξεταστεῖ. Μελετώντας αὐτό τό 3ο μέρος τῆς σειρᾶς, θά πρέπει νά εἶναι δυνατό νά γραφτεῖ ἀκόμα κι ἕνα σχετικά σύνθετο πρόγραμμα.

῾Υπάρχουν μερικοί τρόποι νά είσαχθοῦν πληροφορίες σ᾽ ἕνα ὑπολογιστή. ᾽Εάν πρέπει κάτι νά ὑπολογιστεῖ, οἱ πληροφορίες ἀποτελοῦνται συνήθως ἀπό ἀριθμούς. Στό 2ο μέρος, αὐτοί οἱ ἀριθμοί μπῆκαν στό πρόγραμμα μέσω μεταβλητῶν ἤ εἰσάγοντας τούς ἀριθμούς σάν μέρος τοῦ προγράμματος. ῾Ωστόσο, μιά διαφορετική ἀντιμετώπιση, συχνά θά ἀποδειχτεῖ πιό χρήσιμη, χρησιμοποιώντας μία ἀπό τίς δύο BASIC προτάσεις: INPUT καί READ ... DATA ...

INPUT (ΕΙΣΟΔΟΣ)

Χρησιμοποιώντας τήν πρόταση INPUT, εἰσάγονται πληροφορίες, ἐνῶ ἐκτελεῖται τό πρόγραμμα. ῎Η γιά νά εἴμαστε πιό συγκεκριμένοι: ὅταν ὁ ὑπολογιστής βρεῖ μιά πρόταση INPUT στό πρόγραμμμα, σταματάει καί περιμένει γιά τίς πληροφορίες πού πρέπει νά εἰσαχθοῦν πρίν συνεχίσει μέ τήν ἐκτέλεση τοῦ προγράμματος. ῞Ολη ἡ πρόταση ἀποτελεῖται ἀπό ἕνα ΄INPUT` ἀκολουθούμενο ἀπό τό ὄνομα μιᾶς μεταβλητῆς. Π.χ.:



"Όταν ἐκτελεῖται ἕνα πρόγραμμα, ὁ ὑπολογιστής γράφει ἕνα ἐρωτηματικό μόλις συναντήσει μιά πρόταση INPUT στή γραμμή 10. Μετά περιμένει νά μπεῖ ἕνας ἀριθμός, ἀκολουθούμενος ἀπό τό πλῆκτρο CR. Μόλις μπῆκε ὁ ἀριθμός '256', αὐτή ἡ τιμή δόθηκε στή μεταβλητή A - ὁπότε διεκπεραιώθηκε ὅλο τό πρόγραμμα.

'Η ἴδια πρόταση μπορεῖ νά χρησιμοποιηθεῖ νά δώσει καί σ' ἄλλες μεταβλητές συγχρόνως τιμές: 'INPUT A, B, C, ...'. "Όταν τυπώνεται τό ἐρωτηματικό, πρέπει νά εἰσαχθοῦν οἱ ἀντίστοιχοι ἀριθμοί: ? 123, 62, 23, ...'. 'Η πρόταση INPUT χρησιμοποιεῖται καί γιά νά εἰσάγει ''τιμές κειμένου'' (text values) ὅπως θά ἀναφερθεῖ στό 4ο μέρος. Τό πλεονέκτημα χρησιμοποιήσεως τῆς πρότασης INPUT εἶναι ὅτι δίνει τήν εὐκαιρία νά ἀνοιχτεῖ διάλογος μέ τόν ὑπολογιστή. "Όσο ἀφορᾶ ἐνδιάμεσα ἀποτελέσματα, π.χ., τό πρόγραμμα μπορεῖ νά ἐκτελεστεῖ ἐκ νέου μέ νέες τιμές μέχρι νά ἀποκτηθεῖ ἕνα ἐπιθυμητό τελικό ἀποτέλεσμα. Π.χ. ἄς ὑποθέσουμε ὅτι θέλουμε νά ξέρουμε τούς τόκους μετά ἀπό ὁρισμένα χρόνια (N) καταθέσεως (I) μέ διάφορα ἐπιτόκια (P). 'Η τελική τιμή ἰσοῦται μέ:

$$F = I \times \left(1 + \frac{P}{100}\right)^N$$

Στή BASIC αὐτό γίνεται:

F=1* (1+P/100) ↑ N

"Ένα κατάλληλο πρόγραμμα ἑπομένως εἶναι τό ἑξῆς:

```
> 10 PRINT "ENTER INITIAL INVESTMENT,"
> 11 PRINT "INTEREST RATE"
> 12 PRINT "AND NUMBER OF YEARS"
> 20 INPUT I, P, N
> 30 LET F = I * (1 + P/100) ↑ N
> 40 PRINT "THE FINAL VALUE IS"; F
> 50 END
> RUN
ENTER INITIAL INVESTMENT,
INTEREST RATE
AND NUMBER OF YEARS
? 1000, 9, 10
THE FINAL VALUE IS 2367
>
```

Μ' ἄλλα λόγια μ' ἕνα ἀρχικό ποσό 1000 δρχ. καί ἐπιτόκιο 9%, ἡ τελική τιμή μετά 10 χρόνια θά εἶναι 2367 δρχ.

READ ... DATA ... (ΔΙΑΒΑΣΕ ... ΠΛΗΡΟΦΟΡΙΕΣ ...)

"Ένας ἄλλος τρόπος εἰσόδου πληροφοριῶν εἶναι ἡ χρήση «data bloks», (τεμάχιον πληροφορίας). Τό «data block» εἶναι ἕνα σύνολο πληροφοριῶν, εἰσαγόμενο ἀπό τήν πρόταση DATA. Οἱ διάφοροι ἀριθμοί καί τά κείμενα χωρίζονται μέ κόμματα. "Ένα data block βρίσκεται συνήθως στό τέλος ἑνός προγράμματος, εἰσάγεται πρίν ἀρχίσει ἡ ἐκτέλεση τοῦ προγράμματος. Στό κύριο πρόγραμμα, προτάσεις READ χρησιμοποιοῦνται γιά νά ἀνακαλέσουν τίς πληροφορίες, ὅποτε ἀπαιτεῖται, κάθε καινούργια πρόταση READ ἀνακαλεῖ τόν ἑπόμενο ἀριθμό, κείμενο ἤ σύνολο πληροφοριῶν. Π.χ.

```
> 10 READ A, B, C
> 20 D = A * B * C
> 30 READ E
> 40 F = D/E
> 50 PRINT D, F
> 60 DATA 1, 2, 3, 3
> 70 END
> RUN
6  2
BRK AT 70
>
```

Στή σειρά 10, οἱ τρεῖς πρῶτες πληροφορίες διαβάζονται ἀπό τό data block στή γραμμή 60 καί δίνονται στίς μεταβλητές A, B, καί C. Μέ ἄλλα λόγια τό A γίνεται 1, τό B γίνεται 2 καί τό C γίνεται 3. Μετά τό D ὑπολογίζεται (γραμμή 20), ὁ ἑπόμενος ἀριθμός ἀνακαλεῖται ἀπό τό data block γραμμή 30: τό E γίνεται 3), κ.ο.κ.

"Όπως φαίνεται, μερικές προτάσεις READ μποροῦν νά χρησιμοποιηθοῦν - οἱ πληροφορίες θά διαβαστοῦν μέ σειρά. 'Ομοίως, μποροῦν νά χρησιμοποιηθοῦν μερικές προτάσεις DATA, ὡστόσο αὐτό ἔχει μικρή σημασία - ἁπλῶς χρησιμοποιοῦν-

ται στή σειρά - καί γίνεται πιό δύσκολο νά έντοπιστοῦν καί διευκρινιστοῦν οἱ πληροφορίες ἀργότερα.

Βεβαίως, εἶναι σπουδαῖο ὅτι ἀρκετές πληροφορίες ἀποθηκεύονται στό data block. Ἐάν, μετά τό διάβασμα τῆς τελευταίας πληροφορίας, τό πρόγραμμα συναντήσει κι ἄλλη πρόταση READ, θά τυπώσει κάποιο σχόλιο στίς γραμμές "OUT OF DATA IN xxx", (ὄχι πληροφορίες στή γραμμή xxx, ὅπου xxx ὁ ἀριθμός τῆς γραμμῆς τῆς προτάσεως READ, ὅπου τό data block ἀποδείχτηκε ἄδειο).

Σέ μερικές διαλέκτους BASIC, τά ἴδια περιεχόμενα τῶν data blocks μποροῦν νά ξαναχρησιμοποιηθοῦν μερικές φορές: ἡ πρόταση RESTORE (ἀνα-ἀποθήκευση) κάνει τόν ὑπολογιστή νά ἀρχίσει πάλι στήν ἀρχή τοῦ block.

Οἱ προτάσεις READ ... DATA ... δέν εἶναι γνωστές στή NIBL.

REM (ἐπεξήγηση)

Ἡ πρόταση REM (REMark) προσθέτει ἐπεξηγήσεις στό κείμενο τοῦ προγράμματος, σάν βοήθημα στόν προγραμματιστή. Τό κείμενο ἁπλῶς εἰσάγεται μετά τή πρόταση REM, ἀγνοεῖται ἀπό τό μεταφραστή, ἀλλά ἐπαναεμφανίζεται ὅπου δοθεῖ διαταγή LIST.

Ἡ ἀξία τῆς προτάσεως φαίνεται ἄν δέν ἔχει χρησιμοποιηθεῖ ἕνα πρόγραμμα γιά λίγο καιρό, χρησιμεύει σάν γρήγορος ἐνθυμητής τῆς σημασίας τῶν μεταβλητῶν, κ.λ.π. Π.χ. στό πρόγραμμα ὑπολογισμοῦ τόκων πού χρησιμοποιήθηκε προηγουμένως:

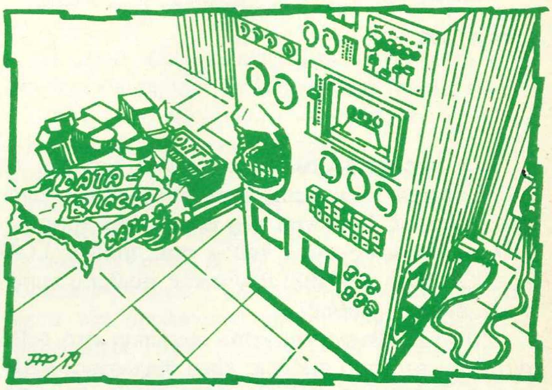
```
> 1 REM THIS PROGRAM CALCULATES
> 2 REM THE FINAL VALUE F
> 3 REM OF AN INITIAL INVESTMENT I
> 4 REM AFTER N YEARS
> 5 REM AT AN INTEREST RATE OF P PERCENT
> 10 PRINT "ENTER INITIAL INVESTMENT,"
> 11 PRINT "INTEREST RATE"
> 12 PRINT "AND NUMBER OF YEARS"
> 20 INPUT I, P, N
> 30 LET F = I * (1 + P/100) ↑ N
> 40 PRINT "THE FINAL VALUE IS"; F
> 50 END
```

Τό ἴδιο τό πρόγραμμα δέν ἐπηρεάζεται ἀπό τίς προτάσεις REM. Ἁπλῶς ξαναπαρουσιάζονται ὅταν ἀπαιτηθεῖ ἕνα listing προγράμματος. Βέβαια, χρησιμοποιοῦν λίγο χῶρο μνήμης - ἀλλά ὑπάρχει ἀρκετός χῶρος διαθέσιμος.

Στάνταρ λειτουργίας

Γιά νά ἁπλουστευτεῖ ὁ προγραμματισμός στή BASIC, ὑπάρχουν 10 στάνταρ μαθηματικές πράξεις διαθέσιμες:

Στάνταρ λειτουργίας	περιγραφή
SIN (x)	sine x
COS (x)	cos x x εἶναι ἡ γωνία σέ ἀκτίνια
TAN (x)	tangent x
ATN (x)	e^x (e = 2,71828...)
EXP (x)	|x| = ἡ ἀπόλυτη τιμή x
ABS (x)	elong |x| = ln |x|, ὁ φυσικός λογάρ.
LOG (x)	$\sqrt{\|x\|}$, ἡ τετραγωνική ρίζα τοῦ |x|
SQR (x)	τό ἀκέραιο μέρος τοῦ x
INT (x)	τό «πρόσημο» τοῦ x»: 1 ἐάν x>0, -1 ἐάν x<0, 0 ἐάν x=0
SEN (x)	



Ἄς σημειωθεῖ ὅτι τό ἡμίτονο, συνημίτονο καί ἐφαπτομένη ἔχουν τή γωνία τους σέ ἀκτίνια (1 ακτ. = 57,2958°, ἤ 1° = 0,017483 rad), τό τόξο τῆς ἐφαπτομένης δίνεται σέ rad. Ὁ LOG (φυσικός λογάριθμος) καί ἡ SQR (τετραγωνική ρίζα) αὐτόματα χρησιμοποιοῦν τήν ἀπόλυτη τιμή τοῦ x, Π.χ.: SQR (-4) = 2.

Ἡ INT λειτουργία προκαλεῖ σύγχυση μερικές φορές. Δίνει τό μεγαλύτερο ἀκέραιο πού εἶναι μικρότερος ἤ ἴσος τοῦ x. Γιά θετικούς αὐτό

σημαίνει νά παραληφτεῖ τό δεκαδικό ἤ κλάσμα: INT (2.78) = 2. Γιά ἀρνητικούς ἀριθμούς ὡστόσο τό ἀποτέλεσμα εἶναι: INT (- 2,78) = - 3! Ὅπως φαίνεται ἀπό τά παραπάνω παραδείγματα, ὁ ἀριθμός (΄x΄) πρέπει νά βρίσκεται πάντα σέ παρένθεση. Συνήθως, ἐπιτρέπεται ἐπίσης ἡ χρήση μεταβλητῆς ἤ ἀλγεβρικῆς ἔκφρασης ἐδῶ. Μερικά παραδείγματα φαίνονται στό παρακάτω πρόγραμμα:

```
> 10 REM EXAMPLES OF STANDARD FUNCTIONS
> 20 PRINT "ENTER ANGLE IN DEGREES"
> 30 INPUT A
> 40 LET B = A/57.2958
> 50 REM B IS ANGLE IN RADIALS
> 60 PRINT "THE SINE OF"; A; "DEGREES IS"; SIN (B)
> 70 PRINT "INT(5.3) = "; INT(5.3)
> 80 PRINT "INT(-8.5)= "; INT(-8.5)
> 90 PRINT "INT(7)= "; INT(7)
> 100 END
> RUN
ENTER ANGLE IN DEGREES
? 30
THE SINE OF 30 DEGREES IS 0.5
INT(5.3)= 5
INT(-8.5)= -9
INT(7)= 7
BRK AT 100
>
```

Δέν πρέπει βέβαια νά προκαλεῖ ἔκπληξη τό ὅτι ὑπάρχουν ἐξαιρέσεις στούς παραπάνω κανόνες. Μερικές διάλεκτοι τῆς BASIC δέν χρησιμοποιοῦν ἀπόλυτες τιμές τοῦ x στίς πράξεις LOG καί SQR (ἐάν εἰσαχθεῖ ἀρνητικός ἀριθμός τυπώνουν ἔνδειξη λάθος).

Στή Ting Basic καί κατά συνέπεια στή NIBL αὐτές οἱ στάνταρ πράξεις εἶναι ἄγνωστες. Μάλιστα ἡ λειτουργία INT θά ἦταν χωρίς σημασία, ἀφοῦ ἡ Tihg Basic ἔτσι κι ἀλλοιῶς ἀναγνωρίζει μόνο ἀκεραίους ἀριμθούς. Ἡ NIBL ἀναγνωρίζει μερικές ἄλλες πράξεις πού θά περιγραφοῦν στό 4ο μέρος.

Προτάσεις Jump

Σ΄ ὅλα τά παραδείγματα προγραμμάτων ὥς τώρα, τά προγράμματα ἐκτελέσθηκαν μέ σταθερή (ἀριθμητική) σειρά. Ἡ πρόταση μέ τό μικρότερο ἀριθμό γραμμῆς ἐκτελέσθηκε πρώτη, κ.ὅ.κ. Ὅπου ἀπαιτεῖται διαφορετική σειρά, «προτάσεις jump» ἀπαιτεῖται διαφορετική σειρά, «προτάσεως jump» μποροῦν νά χρησιμοποιηθοῦν: "GOTO", "IF...THEN..." κ.λ.π.

α) GOTO: Ἡ πρόταση GOTO χρησιμοποιεῖται ὅταν ἕνα «ἅλμα» σέ ὁρισμένη γραμμή ἀπαιτεῖται στό πρόγραμμα. Πρόκειται γιά μιά πρόταση πού δέν χρειάζεται ὁρισμένες προϋποθέσεις γιά νά ἐκτελεστεῖ. Ἕνα παράδειγμα:

```
> 5 N = 0
> 10 PRINT N
> 20 N= N + 1
> 30 GOTO 10
> 40 END
> RUN
0
1
2
3
```

Σ΄ αὐτό τό πρόγραμμα, τό N ἀρχικά εἶναι ἴσο μέ μηδέν καί τυπώνεται αὐτή ἡ τιμή. Στή γραμμή 20, ἡ τιμή αὐξάνεται κατά 1, ἡ γραμμή 30 διατάζει ἕνα ἅλμα πάλι στή γραμμή 10, ὅπου τυπώνεται τό ἀποτέλεσμα (1) κ.ὅ.κ. Ὁ ὑπολογισμός θά ἐξακολουθήσει νά ἐκτελεῖ αὐτή τήν ἐντολή, τυπώνοντας ὅλους τούς ἀριθμούς ἀπό τό 0 κι ἐπάνω - μέχρι νά τελειώσει τό χαρτί ἤ νά φτάσει τό ἀνώτερο ὁριό του.

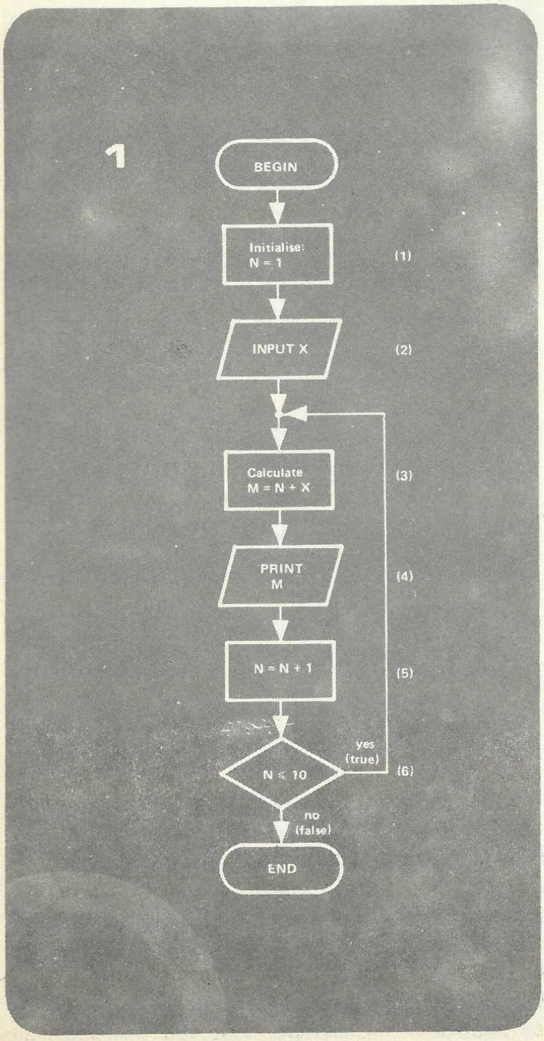
Βέβαια ἄν προκύψει κάτι τέτοιο κατά τήν ἐκτέλεση προγράμματος (λόγω λάθους στόν προγραμματισμό) κάτι πρέπει νά γίνει γιά νά σταματήσει ὁ ὑπολογιστής. Ὑπάρχει: τό πλῆκτρο "BREAK" (σπάω). Μόλις πατηθεῖ αὐτό τό πλῆκτρο, ὁ ὑπολογιστής θά σταματήσει τό πρόγραμμα καί θά τυπώσει κάτι σάν "BRK AT 20" ἤ κάτι παρόμοιο.

β) IF... THEN...(Ἐάν... τότε...): Συχνά, ἕνα ἅλμα σέ νέα γραμμή ἀπαιτεῖται μόνο ὅταν ὁρισμένες συνθῆκες ἱκανοποιηθοῦν. Ἡ ὑποθετική πρόταση δίνει τή πιθανότητα ἐκτέλεσης διάφορων κομματιῶν ἑνός προγράμματος, πού ἐξαρτιῶνται

ἀπό ἐνδιάμεσα ἀποτελέσματα. Γενικά ἡ πρόταση εἰσάγεται ὡς ἑξῆς: IG (ἀντίστοιχη ἔκφραση) THEN (ἀριθμός γραμμῆς).

῾Η ἀντίστοιχη ἔκφραση θά εἶναι κανονικά κάποιο εἶδος σύγκρισης, π.χ. 1«x = 10» ἤ «A>B». ᾿Εάν τό ἀποτέλεσμα τῆς σύγκρισης εἶναι ἀληθές, ὁ ὑπολογιστής θά πάει στή συγκεκριμένη γραμμή, ἐάν ὄχι, ἁπλῶς θά συνεχίσει στήν ἑπόμενη γραμμή.

Σάν παράδειγμα, ἄς ὑποθέσουμε ὅτι ἀπαιτεῖται ἕνα πρόγραμμα πού θά τυπώσει τό πολλαπλασιαστικό πίνακα ὁποιουδήποτε ἀριθμοῦ (΄x`). Τό διάγραμμα ροῆς (flow chart) γιά ἕνα κατάλληλο πρόγραμμα θά ἦταν σάν αὐτό πού δείχνεται στό παρακάτω σχῆμα:



Τό ἀντίστοιχο πρόγραμμα εἶναι τό ἀκόλουθο:

```
> 10 REM THIS PROGRAM CALCULATES
> 15 REM AND PRINTS 1xX THROUGH 10xX
> 20 LET N= 1
> 30 PRINT "ENTER X"
> 40 INPUT X
> 50 LET M = N * X
> 60 PRINT N; "*"; X; "="; M
> 70 LET N= N+1
> 80 IF N<= 10 THEN 50
> 90 END
> RUN
ENTER X
? 12
1 * 12 = 12
2 * 12 = 24
:
: (etc.)
:
10 * 12 = 120
BRK AT 90
>
```

῾Η σειρά τῶν πράξεων εἶναι ἡ ἀκόλουθη: μετά ἀπό τήν διαδικασία τοποθετήσεως (initalisation) - σ᾿ αὐτή τήν περίπτωση, κάνοντας N = 1 - ὁ ὑπολογισμός θά ἀπαιτήσει τή τιμή τοῦ x (βῆμα 2). Στά βήματα 3 καί 4, τό 1×X ὑπολογίζεται καί τυπώνεται, μετά τό ὁποῖο τό N αὐξάνεται κατά 1 (βῆμα 5). Μετά, στό βῆμα 6, ἡ τιμή τοῦ N ἐλέγχεται ἐάν εἶναι ≤ 10, ὁ ὑπολογιστής πρέπει νά γυρίσει στό βῆμα 3 γιά τόν ἑπόμενο ὑπολογισμό. Μετά τόν ὑπολογισμό καί τό τύπωμα ὅλων τῶν τιμῶν μέχρι 10×X, τό N θά γίνει 11 στό βῆμα 5, τό ἀποτέλεσμα τῆς σύγκρισης στό βῆμα 6 θά εἶναι ψευδές καί τό πρόγραμμα θά τελειώσει.

Σέ μερικές διαλέκτους BASIC (καί τή NIBL), εἶναι δυνατό νά μπεῖ μιά πρόταση μετά τό THEN, ἀντί γιά μετά ἀριθμό γραμμῆς. Τότε, ἐάν τό ἀποτέλεσμα τῆς σύγκρισης εἶναι ἀληθές, ἡ πρόταση θά ἐκτελεστεῖ, ἀλλοιῶς θά ἀγνοηθεῖ. Στή NIBL, ἐάν ἕνα ἅλμα σέ γραμμή ἀπαιτηθεῖ, τό "GOTO" πρέπει νά χρησιμοποιηθεῖ ἀντί γιά τό "THEN". Οἱ δύο πιθανότητες εἶναι στή NIBL οἱ ἑξῆς:

IF (ἀντίστοιχη πρόταση) GOTO (ἀριθμό γραμμῆς), καί

IF (ἀντίστοιχη πρόταση) THEN (πρόταση).

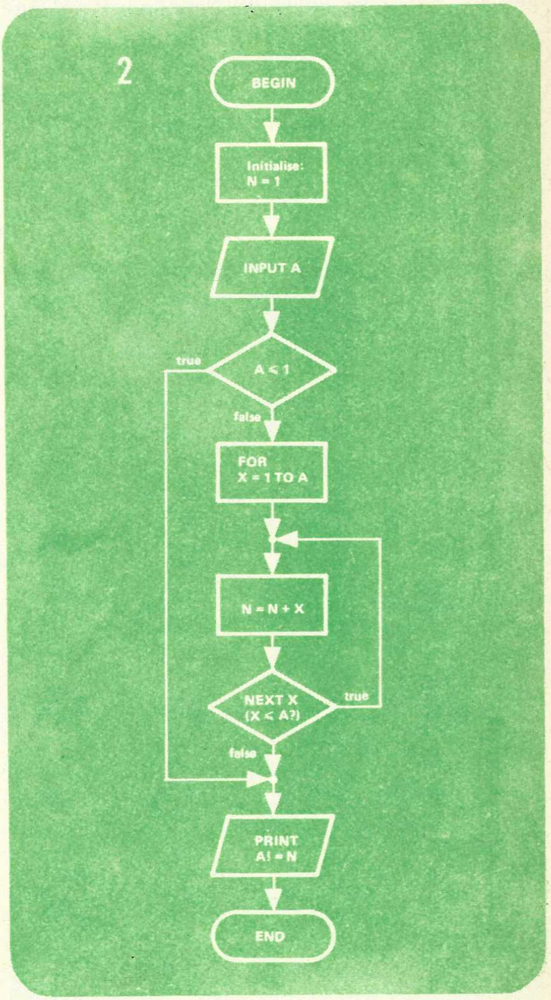
γ) FOR ... NEXT ... (γιά ... μετά): Ἕνας ἄλλος

τρόπος νά ἐκτελεστεῖ τό ἴδιο κομμάτι τοῦ προγράμματος μερικές φορές διαδοχικά, εἶναι μέ τήν πρόταση FOR ... NEXT... Στήν πραγματικότητα πρόκειται γιά δύο προτάσεις πού πρέπει νά εἰσαχθοῦν σέ διαφορετικές γραμμές τοῦ προγράμματος, ὅπως φαίνεται στό πάρα κάτω παράδειγμα:

```
> 10 FOR A= 1 TO 5
> 20 PRINT A↑2
> 30 NEXT A
> 40 END
> RUN
1
4
9
16
25
BRK AT 40
>
```

Στήν πρόταση FOR, μιά μεταβλητή (A), μιά ἀρχική τιμή (1) καί μιά τελική τιμή (5) δίνονται. Τό τμῆμα τοῦ προγράμματος μεταξύ τῶν προτάσεων FOR καί NEXT ἐκτελεῖται μερικές φορές ἀρχίζοντας μέ τήν ἀρχική τιμή τοῦ A καί αὐξάνοντάς την κατά 1 μέχρι νά φθάσουμε στή τελευταία τιμή. Ἡ ἀρχική καί τελική τιμή δέν εἶναι ἀνάγκη νά δίνονται σάν ἀριθμοί. Καί μεταβλητές καί ἐκφράσεις μποροῦν νά χρησιμοποιηθοῦν (π.χ. "FOR A = N TO SO * N"). Ἡ τελική τιμή πρέπει βέβαια νά εἶναι μεγαλύτερη ἤ ἴση μέ τήν ἀρχική. Δέν ἐνδείκνυται νά χρησιμοποιηθεῖ ἡ μεταβλητή. Π.χ. A στό παραπάνω παράδειγμα σέ ὅλα τά σημεῖα τοῦ προγράμματος - ἐκτός ἀπό ἄλλες προτάσεις FOR. Οἱ προτάσεις μεταξύ FOR καί NEXT ἀναφέρονται συχνά σάν "FOR - NEXT block". Πρίν ἐκτελέσει ὁ ὑπολογιστής ἕνα τέτοιο block, πρῶτα ὑπολογίζει τίς ἀρχικές καί τελικές τιμές γιά τή μεταβλητή. Ἐάν αὐτές οἱ τιμές ἐξαρτῶνται ἀπό ἄλλες μεταβλητές, ἡ τιμή ὑπολογίζεται ἐκείνη τή στιγμή. Ὅταν ὑπολογιστοῦν οἱ ἀρχικές καί τελικές τιμές παραμένουν ἀμετάβλητες καθώς ἐκτελεῖται τό block.

Ἕνα πιό «πρακτικό» παράδειγμα τῶν προτάσεων FOR... NEXT... εἶναι ἕνα πρόγραμμα γιά τόν ὑπολογισμό τοῦ A! (A παραγοντικό). (A! = 1* 2* 3*....* (A-2)* (A-1)* A, π.χ. 3! 1* 2* 3 = 6. Ἐξ ὁρισμοῦ 0! = 1). Ἕνα flow chart γιά ἕνα κατάλληλο πρόγραμμα δίνεται πάρακατω:



Μετά ἀπό τή διαδικασία θέσεως τιμῶν, εἰσάγεται ἡ τιμή A (ἕνας θετικός ἀριθμός). Ἐάν A = 0 ἤ 1, τό ἀποτέλεσμα θά εἶναι 1 καί δέν ὑπάρχει λόγος νά ἐκτελεσθεῖ ὅλο τό πρόγραμμα - μάλιστα γιά A = 0, δέν δουλεύει ἀφοῦ ἡ τελική τιμή γιά τήν πρόταση FOR θά ἦταν μικρότερη ἀπό τήν ἀρχική τιμή - ἔτσι τό ἀποτέλεσμα μπορεῖ νά τυπωθεῖ ἀμέσως. Γιά ὅλες τίς ἄλλες τιμές τοῦ A, ἐκτελεῖται ὁ ὑπολογισμός πού περιγράφηκε πα-

ραπάνω καί τυπώνεται τό ἀποτέλεσμα. Τό ἀνάλογο πρόγραμμα εἶναι τό ἑξῆς:

```
> 10 REM CALCULATION OF A!
> 20 LET N= 1
> 30 PRINT "ENTER A"
> 40 INPUT A
> 50 IF A<=1 THEN 90
> 60 FOR X= 1 TO A
> 70 LET N = N * X
> 80 NEXT X
> 90 PRINT A; "! ="; N
> 100 END
> RUN
ENTER A
? 3
3! = 6
BRK AT 100
>
```

Μετά ἀπό τό γράψιμο ἑνός τέτοιου προγράμματος, πρέπει νά ἐλεγχθεῖ μέ μερικές τιμές γιά τίς μεταβλητές πού δίνουν γνωστά ἀποτελέσματα (σ' αὐτό τό παράδειγμα, ἔστω τό 3 καί τό 5) εἰδική προσοχή πρέπει νά δοθεῖ σέ τιμές πού ἀπαιτοῦν διαφορετική διαδικασία (0 καί 1 σ' αὐτό τό πρόγραμμα).

δ) FOR .... TO....STEP... Στίς προτάσεις FOR... NEXT....ὅπως περιγράφτηκε ὥς ἐδῶ, ἡ μεταβλητή αὐξάνεται κατά 1 κάθε φορά πού ἀρχίζει ἐκ νέου ἡ ἐκτέλεση. Αὐτό δέν εἶναι πάντα ἐπιθυμητό: ἴσως ἀπαιτεῖται μεγαλύτερη ἤ μικρότερη αὔξηση. Σ' αὐτές τίς περιπτώσεις ἡ πρόταση STEP προστίθεται ὡς ἀκολούθως: FOR 1 = - 90, TO STEP 15.

Ἡ μεταβλητή θά αὐξάνεται τώρα 1 κατά 15. Καθορίζονται κι ἄλλα βήματα, ἀκόμα καί ἀρνητικά, ὁπότε ἡ τιμή μειώνεται. Π.χ.: FOR 1=0 TO-90 STEP-10.

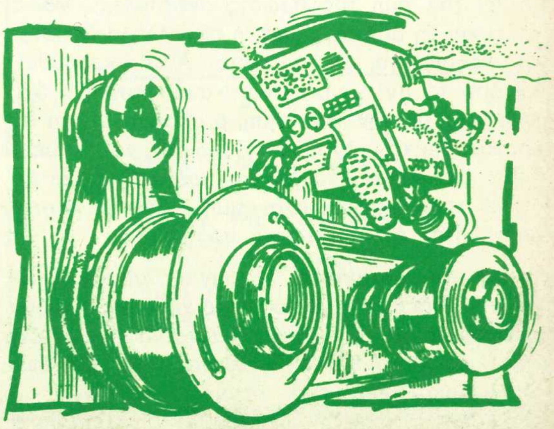
Πρέπει νά δοθεῖ προσοχή ὥστε ἡ μεταβλητή νά φθάσει στή τελική τιμή μέ τά δεδομένα βήματα. Θά ἦταν, π.χ. πολύ ἐπικίνδυνο νά δοθεῖ ἡ ἐντολή: FOR 1=0 TO 90 STEP-10

ε) DO ... UNTIL ....(ΚΑΝΕ... ΜΕΧΡΙ...). Ἡ πρόταση αὐτή εἶναι γνωστή σέ πολύ λίγες διαλέκτους BASIC. Ὁ λόγος πού θά περιγραφεῖ εἶναι ὅτι ἡ πρόταση εἶναι γνωστή καί στή NIBL. Σάν παράδειγμα τῆς χρήσης της, ἄς θεωρήσουμε τό παρακάτω τμῆμα προγράμματος:

```
> 170 N= 0
> 180 DO
> 190   N= N+1
> 200   PRINT "EXAMPLE OF A DO - UNTIL LOOP"
> 210 UNTIL N= 5
> 220 . . .
```

Οἱ προτάσεις μεταξύ DO καί UNTIL ἐπαναλαμβάνονται γιά ὅσο ἡ ἔκφραση μετά τό UNTIL παραμένει ψευδής. Μόλις γίνει ἀληθής, ὁ ὑπολογισμός συνεχίζει μέ ὅλες τίς προτάσεις τοῦ προγράμματος. Στό προηγούμενο παράδειγμα, τό κείμενο θά τυπωθεῖ 5 φορές.

Ὑπολειτουργίες, GOSUB.... RETURN. Ὅταν γράφονται μεγάλα προγράμματα, «ὑπολειτουργίες» (subroutines) μποροῦν νά εἶναι ἐξαιρετικά χρήσιμες. Πολλές φορές συμβαίνει ἕνα ὁρισμένο κομμάτι τοῦ προγράμματος νά ἀπαιτεῖται μερικές φορές, π.χ. ἕνας σύνθετος ὑπολογισμός ὅπως ὁ A! Αὐτό τό τμῆμα τότε εἰσάγεται μετά τό βασικό πρόγραμμα, σάν μιά ὑπολειτουργία (subroutine).



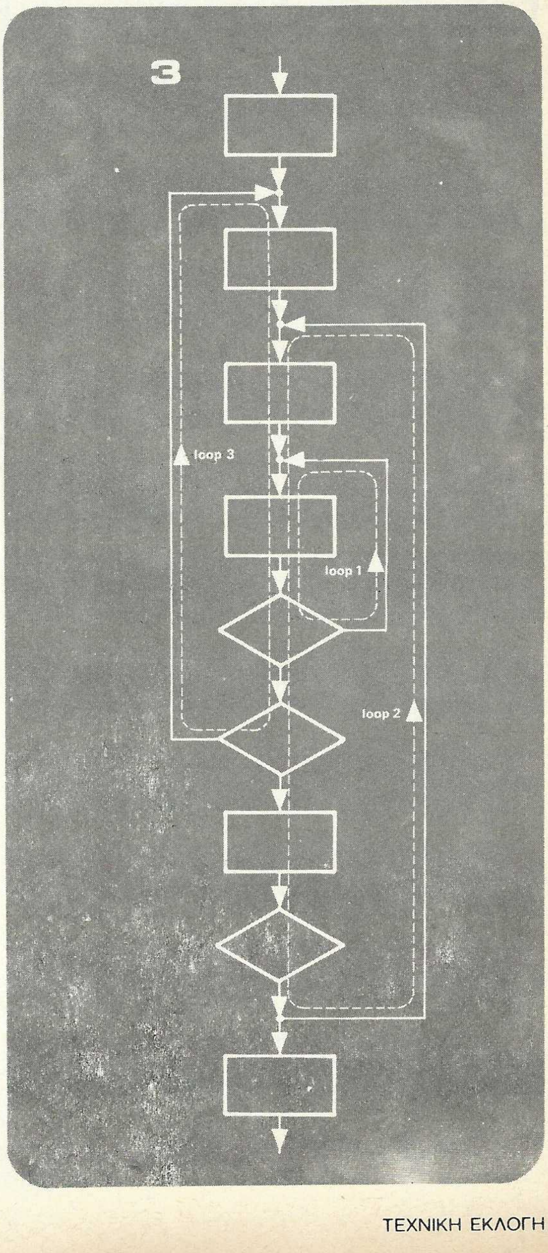
῞Οταν ἀπαιτεῖται μιά πρόταση GOSUB, ἀκολουθούμενη ἀπό τόν πρῶτο ἀριθμό γραμμῆς τῆς subroutine. Τό ἴδιο τό subroutine κανονικά τελειώνει μέ ὁδηγία RETURN (ἐπέστρεψε) ἀναγκάζοντας τόν ὑπολογιστή νά ἐπιστρέψει στόν ἑπόμενο ἀριθμό γραμμῆς. Σάν παράδειγμα, ἄς ὑποθέσουμε ὅτι ὁ ὑπολογιστής χρησιμοποιεῖται σάν βοήθημα στή σχεδίαση κυκλώματος καί ὅτι κανονικά θά χρησιμοποιηθοῦν στό κύκλωμα δύο ἐν παραλλήλῳ ἀντιστάσεις γιά νά πλησιάσουμε μιά ἐπιθυμητή τιμή ἀντιστάσεως. ῞Ενα τμῆμα τοῦ βασικοῦ προγράμματος καί τοῦ subroutine μπορεῖ νά εἶναι τό ἑξῆς:

```
> 110 PRINT "SELECT TWO RESISTANCE VALUES FOR R21"
> 120 GOSUB 500
> 130 A= P
:
> 500 REM SUBROUTINE FOR CALCULATING PARALLEL RESISTANCE"
> 510 PRINT "ENTER VALUE FIRST RESISTOR"
> 520 INPUT X
> 530 PRINT "ENTER VALUE SECOND RESISTOR"
> 540 INPUT Y
> 550 P= X•Y/(X+Y)
> 560 PRINT "THE PARALLEL RESISTANCE IS"; P
> 570 PRINT "IS THIS VALUE SATISFACTORY?";
> 580 PRINT "IF SO, ENTER 1; IF NOT, ENTER 0"
> 590 INPUT Z
> 600 IF Z= 0  THEN 510
> 610 RETURN
```

Οταν ὁ ὑπολογιστής φθάσει στό σημεῖο τοῦ βασικοῦ προγράμματος ὅπου χρειάζεται τήν τιμή τοῦ R21- πού θά πλησιασθεῖ τοποθετώντας παράλληλα δύο ἀντιστάσεις - τυπώνει τήν ἀντίστοιχη ἐντολή καί πηγαίνει μετά στό subroutine. Ἐκεῖ, ἡ ἐντολή γίνεται πιό συγκεκριμένη ἐπειδή ζητάει τήν τιμή τῆς πρώτης ἀντίστασης. Μόλις λάβει καί τή δεύτερη τιμή, ἡ παράλληλη ἀντίσταση ὑπολογίζεται καί τυπώνεται. Αὐτό ἀκολουθεῖται ἀπό τό ἄν ἡ τιμή εἶναι ἱκανοποιητική ἤ ὄχι. Ἀνάλογα μέ τήν ἀπάντηση, ὁ ὑπολογιστής ἤ θά ἐπαναλάβει τή subroutine (ζητώντας νέες τιμές) ἤ θά ἐπανέλθει στό βασικό πρόγραμμα καί θά δώσει τήν ὑπολογισμένη τιμή στήν ἀντίστοιχη μεταβλητή (A) στό βασικό πρόγραμμα.

Loops προγραμμάτων. ῎Εχουν περιγραφεῖ μερικές προτάσεις πού μποροῦν νά χρησιμοποιηθοῦν ὅταν ἀπαιτεῖται loop. Εἶναι δυνατό νά ἔχουμε ἕνα loop μέσα στό ἄλλο - μάλιστα στό παραπάνω πρόγραμμα εἶναι ἕνα τέτοιο παράδειγμα: τό subroutine εἶναι ἕνα loop ἄν μέσα στό subroutine ὑπάρχει μιά πρόταση IF....THEN.... τότε δημιουργεῖται ἕνα loop περισσότερο.

Τό σχεδιάγραμμα Νο 3 δείχνει loops μέσα σέ loops σέ ἕνα flow chart. Σ᾽ αὐτή τήν περίπτωση, ὡστόσο, τό loop εἶναι ἐπικίνδυνο. Ἐάν τό loop εἶναι ἤ μία FOR - NEXT ἤ DO-UNTIL loop, ἅλμα σέ ἀριθμό γραμμῆς ἐκτός ἀπό τό loop δέν ἐπιτρέπεται: ὁ ὑπολογιστής πρέπει πρῶτα νά ἔχει τήν εὐκαιρία νά τελειώσει τήν καταμέτρηση (τελική τιμή σέ μιά πρόταση FOR - NEXT) ἤ νά



ἐκπληρώσει τή συνθήκη πού ὁρίζεται στήν πρόταση DO-UNTIL.

Ἀνάλογα μέ τή διάλεκτο τῆς BASIC, ὑπάρχει ἕνας ἀνώτερος ἀριθμός loops πού δέν πρέπει νά ξεπερνιέται. Στή NIBL, π.χ., τά περισσότερα loops εἶναι:

GOSUB - RETURN - 8
FOR - NEXT - 4
DO - UNTIL - 8

Ἐάν ὁ ὑπολογιστής δεῖ λάθη στά loops, θά τυπώσει κάποιο κατάλληλο σχόλιο. Στή NIBL, π.χ., τά σχόλια εἶναι ὡς ἑξῆς:

Print-out:	Νόημα
NEST ERROR:	Ὑπάρχουν πολλά loops μέσα σέ loops.
FOR ERROR:	Τό FOR δέν ἀκολουθεῖται ἀπό τό NEXT.
NEXT ERROR:	Τό NEXT δέν προηγεῖται τοῦ FOR.
NOGO ERROR:	Ὁ ἀριθμός γραμμῆς πού δείχνεται σέ μιά πρόταση GOTO ἤ GOSUB δέν ὑπάρχει.
RTRN ERROR:	Τό RETURN δέν ὑπάρχει μετά τό GOSUB.
UNTIL ERROR:	Τό DO δέν προηγήθηκε τοῦ UNTIL.

Παράδειγμα προγράμματος

Γιά νά φανοῦν οἱ δυνατότητες τῆς θεωρίας BASIC πού μελετήθηκαν ὥς τώρα, ἄς θεωρήσουμε ἕνα πρόγραμμα πού μπορεῖ νά ὑπολογίσει τό κόστος πού ὑπάρχει σέ μιά ἀγορά. Θά λάβουμε:

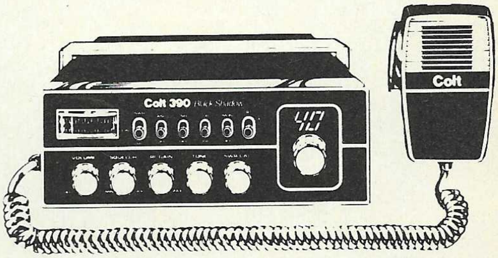
- Ἕνα σταθερό ποσό πληρώνεται κάθε μῆνα, πού ἔχει καί τόκο: τό μηνιάτικο M. Αὐτό τό ποσό πρόκειται νά ὑπολογισθεῖ.
- Ἡ τιμή κόστους τοῦ ἀντικειμένου πού θέλουμε νά ἀγοράσουμε εἶναι C.
- Ἕνα ἀρχικό κεφάλαιο, S, εἶναι διαθέσιμο.
- Ἕνα δάνειο, L, ἀπαιτεῖται: L = C-S.
- Τό ἐπιτόκιο εἶναι P(%).
- Τό δάνειο πρέπει νά ξεπληρωθεῖ σέ N χρόνια.

Θά ὑπολογισθοῦν ἐπίσης:

- Τό ποσό πού πληρώθηκε (R) μετά X χρόνια.
- Τό χρωστούμενο (O) μετά ἀπό X χρόνια.
- Ὁ συνολικός τόκος τοῦ ἀντικειμένου ἔχει πληρωθεῖ μετά N χρόνια.

Σέ σύνολο ἀπαιτοῦνται οἱ ἑξῆς μεταβλητές:
Ἡ τιμή κόστους C;
Τό ἀρχικό κεφάλαιο, S;
Τό ἐπιτόκιο, P;

ΡΑΔΙΟΤΗΛΕΦΩΝΑ
CB



τό CB πού δίνει ρέστα

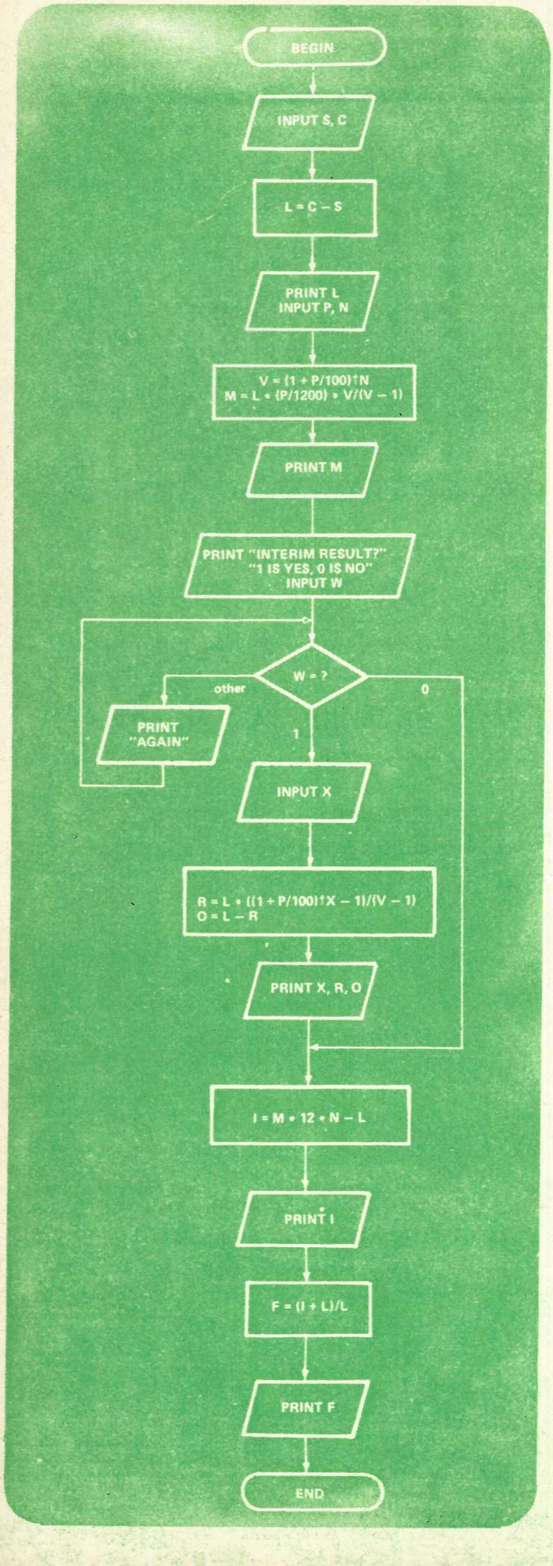
Στήν ἀποθήκη μας θά βρεῖτε:

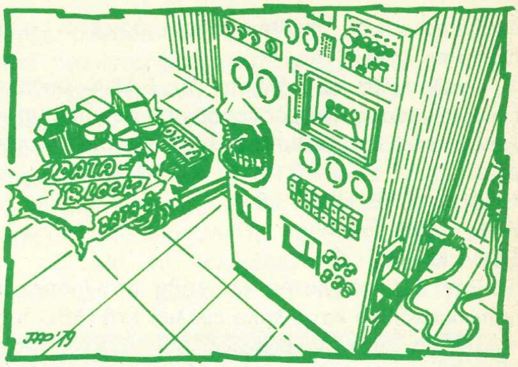
- στερεοφωνικά συγκροτήματα
- ἠχεῖα αὐτοκινήτων
- στερεοφωνικά ἠχεῖα
- Μεγάφωνα
- Ἀντένες CB
- ραδιομαγνητόφωνα καί κασσετόφωνα αὐτοκινήτων 16W
- κομπιοῦτερ κ.λ.π.

ΚΕΝΤΡΙΚΗ ΔΙΑΘΕΣΗ-ΧΟΝΔΡΙΚΗ ΠΩΛΗΣΗ

ΚΩΝ. ΚΑΡΟΥΜΠΑΛΗΣ

ΑΓ. ΔΗΜΗΤΡΙΟΥ ΟΠΛΩΝ 23
ΑΘΗΝΑ (218) - ΤΗΛ. 8318708





῾Ο συνολικός ἀριθμός χρόνων, N; Βασισμένοι σ᾽ αὐτές τίς πληροφορίες, θά ὑπολογισθοῦν τά ἑξῆς:

- Τό δάνειο, L = C - S;
- ῾Η μηνιαία δόση,

$$M = \frac{1}{12} \cdot L \cdot \frac{P}{100} \cdot \frac{(1 + \frac{P}{100})^N}{(1 + \frac{P}{100})^N - 1},$$

- Τό ποσό πού πληρώνεται μετά ἀπό X χρόνια,

$$R = L \cdot \frac{(1 + \frac{P}{100})^X - 1}{(1 + \frac{P}{100})^N - 1};$$

- Τό ὑπολειπόμενο μετά X χρόνια, O = L - R.
- ῾Ο συνολικός τόκος πού πληρώθηκες σέ N χρόνια,
  I = M . 12 . N - L,
- Πόσες φορές πληρώθηκε τό κόστος F=I+L/L.

Σ᾽ αὐτούς τούς ὑπολογισμούς, τό

$$(1 + \frac{P}{100})^{\cdot}$$

παρουσιάζεται μερικές φορές.

Εἶναι καλή ἰδέα νά ὑπολογισθεῖ αὐτό στήν ἀρχή τοῦ προγράμματος καί νά δοθεῖ ἡ τιμή πού βρίσκεται σέ μιά νέα μεταβλητήν. Οἱ διάφοροι τύποι θά εἶναι τότε:

L = C - S, V = (1+P/100)↑ N,
M = L* (P/1200)* V/(v-1), R = L* (1+P/100)↑(x-1)/(v-1),
O = L-R, I = M* 12* N-L, F = (1+L)/L.

῎Ετσι σχηματίζεται τό ἑξῆς διάγραμμα ροῆς καί πρόγραμμα:

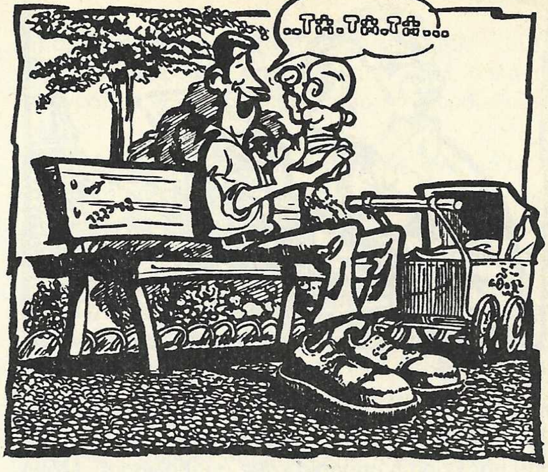
```
> 10 PRINT "ELEKTOR SOFTWARE SERVICE"
> 20 PRINT "LOAN REPAYMENT"
> 30 PRINT "WHAT STARTING CAPITAL IS AVAILABLE?"
> 40 INPUT S
> 50 PRINT "WHAT IS THE COST PRICE?"
> 60 INPUT C
> 70 LET L = C – S
> 80 PRINT "THE NECESSARY LOAN IS"; L; "POUNDS"
> 90 PRINT "WHAT IS THE INTEREST RATE?"
> 100 INPUT P
> 110 PRINT "IN HOW MANY YEARS MUST THE LOAN BE REPAYED?"
> 120 INPUT N
> 130 LET V = (1 + P/100) ↑N
> 140 LET M = L * (P/1200) * V / (V – 1)
> 150 PRINT "THE MONTHLY PAYMENT IS"; M; "POUNDS"
> 160 PRINT "DO YOU WISH TO KNOW THE AMOUNTS ALREADY REPAYED"
> 170 PRINT "AND STILL OUTSTANDING AFTER A NUMBER OF YEARS?"
> 180 PRINT "PLEASE ENTER 1 FOR YES, OR 0 FOR NO"
> 190 INPUT W
> 200 IF W=0 THEN 320
> 210 IF W=1 THEN 250
> 220 PRINT "PLEASE STUDY THE KEYBOARD CAREFULLY, THEN"
> 230 PRINT "ENTER 1 FOR YES, OR 0 FOR NO. TRY AGAIN"
> 240 GOTO 190
> 250 PRINT "AFTER HOW MANY YEARS?"
> 260 INPUT X
> 270 LET R = L * ((1 + P/100)↑X – 1) / (V – 1)
> 280 LET 0 = L – R
> 290 PRINT "AFTER"; X; "YEARS"
> 300 PRINT "YOU WILL HAVE REPAYED"; R; "POUNDS"
> 310 PRINT "AND"; 0; "POUNDS WILL REMAIN OUTSTANDING"
> 320 LET I = M * 12 * N – L
> 330 PRINT "AFTER"; N; "YEARS, YOU WILL HAVE PAYED"
> 340 PRINT "A TOTAL OF"; I; "POUNDS INTEREST"
> 350 LET F = (I + L)/L
> 360 PRINT "YOU WILL HAVE PAYED"; F; "TIMES THE ORIGINAL LOAN"
> 370 END
```

Οἱ γραμμές 220, 230 καί 240 περιέχονται στό πρόγραμμα μήπως δοθεῖ ἀδύνατη ἀπάντηση (π.χ. ´5´) ἀντί γιά 0 ἤ 1. ᾿Αφοῦ καμμιά ἀπό τίς συνθῆκες πού δίνεται στίς γραμμές 200 καί 210 ἱκανοποιοῦνται, τότε, ὁ ὑπολογιστής θά προειδοποιήσει τό χειριστή καί θά ἐπιστρέψει στή γραμμή 190 γιά νέα δοκιμή. ᾿Αφοῦ μπεῖ τό πρόγραμμα, μπορεῖ νά δοκιμασθεῖ:

(Συνεχίζεται)

```
> RUN
ELEKTOR SOFTWARE SERVICE
LOAN REPAYMENT
WHAT STARTING CAPITAL IS AVAILABLE?
? 0
WHAT IS THE COST PRICE?
? 10000
THE NECESSARY LOAN IS 10000 POUNDS
WHAT IS THE INTEREST RATE?
? 9
IN HOW MANY YEARS MUST THE LOAN BE REPAYED?
? 30
THE MONTHLY PAYMENT IS 81 POUNDS
DO YOU WISH TO KNOW THE AMOUNTS ALREADY REPAYED
AND STILL OUTSTANDING AFTER A NUMBER OF YEARS?
PLEASE ENTER 1 FOR YES, OR 0 FOR NO
? 2
PLEASE STUDY THE KEYBOARD CAREFULLY, THEN
ENTER 1 FOR YES, OR 0 FOR NO. TRY AGAIN
? 1
AFTER HOW MANY YEARS?
? 10
AFTER 10 YEARS
YOU WILL HAVE REPAYED 1114 POUNDS
AND 8886 POUNDS WILL REMAIN OUTSTANDING
AFTER 30 YEARS, YOU WILL HAVE PAYED
A TOTAL OF 19160 POUNDS INTEREST
YOU WILL HAVE PAYED 2.916 TIMES THE ORIGINAL LOAN
```

BASIC η απλη γλωσσα των υπολογιστων



ΜΕΡΟΣ 4

Γράφει ὁ
Μ. ΟΙΚΟΝΟΜΑΚΗΣ

Στά πρῶτα τρία κομμάτια αὐτῆς τῆς σειρᾶς, ἀσχοληθήκαμε γενικά μέ τίς δυνατότητες τοῦ συστήματος BASIC, σά γλώσσα προγραμματισμοῦ γιά τούς ὑπολογιστές. Σέ αὐτό τό τελευταῖο μέρος, θά περιγραφοῦν μερικές ἀξιοσημείωτες παρατηρήσεις, πάνω στό «ἐκτεταμένο BASIC» καί στό NIBL. Πρῶτα ὅμως θά δοθοῦν μικρές χρήσιμες πληροφορίες σχετικά μέ τό προγραμματισμό - καθώς καί πληροφορίες στό «ξεκόλλημα» προγραμμάτων.

Χρήσιμες μικροπληροφορίες προγραμματισμοῦ

Τά προγράμματα, γενικά, ἀπαιτοῦνται γιά νά κάνουν κάποιο ὑπολογισμό, νά ρυθμίζουν ἕνα σύστημα, ἤ γιά κάποια παρόμοια δουλειά. Τό πρῶτο πού πρέπει νά γίνει - πρίν ἀκόμα σκεφθεῖ κανείς τό ἴδιο τό πρόγραμμα - εἶναι νά καθοριστεῖ προσεκτικά τό πρόβλημα. Γιά ἕνα ὑπολογισμό, π.χ., εἶναι σπουδαῖο νά εἶναι γνωστές περίπου οἱ τιμές εἰσόδου, ἐάν μποροῦν ἤ ὄχι νά εἶναι θετικές, ἀρνητικές, μηδέν, κ.λ.π.

Ἐάν ἀπαιτεῖται κάποια μορφή ρυθμιστικοῦ συστήματος, εἶναι ἀπαραίτητο νά εἶναι γνωστό μέ ποιά σειρά πρέπει νά γίνουν διάφορες πράξεις καί τί μπορεῖ νά πάει ἄσχημα! Προβλήματα κατά τήν ἐκτέλεση προγράμματος, εἶναι συχνά ἀποτέλεσμα ἀτελοῦς ἤ ἀνακριβοῦς ὁρισμοῦ τῆς δουλειᾶς πού πρέπει νά ἐκτελεστεῖ. Μόλις διασαφηνισθεῖ ἡ πράξη, πρέπει νά φτιαχτεῖ ἕνα διάγραμμα ροῆς. Ἔτσι δίδεται σαφής εἰκόνα τῆς βασικῆς δομῆς τοῦ προγράμματος: πιθανές ἁπλοποιήσεις, διορθώσεις ἤ τροποποιήσεις, συχνά γίνονται ἔτσι προφανεῖς. Μερικές φορές ἀνακαλύπτεται ὅτι τό πρόγραμμα μπορεῖ νά ἁπλοποιηθεῖ αἰσθητά μέ μικρή τροποποίηση τοῦ ὁρισμοῦ τῆς λειτουργίας. Δέν πρέπει κανείς νά περιμένει θαύματα ἀπό τόν ὑπολογιστή - ὁποιοδήποτε προγραμματικό διάβημα θά ἀποτύχει ἐάν τό πρόβλημα δέν γίνει πλήρως κατανοητό (ἤ ἡ πράξη περιγραφεῖ τελείως), πρίν ἀρχίσει ἡ δημιουργία τοῦ προγράμματος.

Ὅταν ἀρχίσει ἡ καταγραφή τοῦ ἴδιου τοῦ προγράμματος, εἶναι καλή ἰδέα νά ἀρχίσει κανείς μέ διαδικασία «ἀντιστοιχίσεως γραμμάτων» - ὅλες οἱ μεταβλητές παίρνουν μιά ἀρχική τιμή (συνήθως 0 ἤ 1), μέσω τῆς προτάσεως LET. Δίνοντας στίς μεταβλητές τιμή μηδέν, φαίνεται ἄχρηστο πολλές φορές, ἀφοῦ γίνεται αὐτόματα ὅταν ἀνάβεται ὁ ὑπολογιστής. Ὡστόσο, αὐτό δέν συμβαίνει πάντοτε καί οἱ μεταβλητές μπορεῖ νά ἔχουν μιά τιμή πού τούς δόθηκε σέ παλιότερο προγραμματισμό. Γενικά, ἡ παραπάνω διαδικασία πρέπει νά θεωρεῖ-

ται ὑποχρεωτική. Ἐάν χρησιμοποιοῦνται ἀρκετές μεταβλητές εἶναι καλή ἰδέα νά καταγράφεται κά- που πρόχειρα ἡ σημασία κάθε συμβολισμοῦ. Μόνο ἀρκετά ἔμπειροι προγραμματιστές πρέπει νά ἀ- σχολοῦνται μέ μεγάλα καί πολύπλοκα προγράμ- ματα.

Πολύ συχνά, μεγάλα προγράμματα μποροῦν νά «κοποῦν» σέ μερικά μικρότερα τμήματα. Καθέ- να ἀπ' αὐτά ἐλέγχεται χωριστά, καί ὅταν δου- λεύουν ὅλα καλά, μποροῦν νά «κολληθοῦν» μαζί, σχηματίζοντας τό τελειοποιημένο τελικό πρόγραμ- μα.

Μόλις γραφτεῖ τό πρόγραμμα, εἶναι ὥρα γιά τή τελική δοκιμή. Αὐτό πού θά διαπιστωθεῖ εἶναι πώς εἶναι πράγματι σπάνιο νά δουλέψει σωστά ἕνα πρόγραμμα ἀπό τήν πρώτη φορά. Ἑπομένως τό ἑπόμενο βῆμα εἶναι τό «ξεκόλλημα» (ἤ ἐπιδιόρ- θωση) τοῦ προγράμματος.

Πιθανά λάθη μπορεῖ νά εἶναι δύο κατηγοριῶν: αὐτά πού ἀνακαλύπτονται ἀπό τόν ἴδιο ὑπολογι- στή (λάθη διαδικασίας) καί ἐκεῖνα πού γίνονται προφανῆ μόνο ἀπό λάθος ἐκτέλεσης τοῦ προ- γράμματος (ἐκτελεστικά λάθη).

Τά πρῶτα συνήθως ἀνακαλύπτονται ἀπό τό μεταφραστή, ὅταν ἐπιχειρεῖ νά μεταφράσει τίς ὁδηγίες σέ μηχανική γλώσσα. Κατά παράδοξο τρόπο, οἱ ὁδηγίες δέν συμφωνοῦν μέ τούς κανό- νες τοῦ BASIC. Συνήθως εἶναι «ἀνόητα» λάθη – ἀλλοιῶς ὁ ὑπολογιστής δέν θά τά ἀνακάλυπτε – ὅπως τυπώνοντας PRANT ἀντί γιά PRINT, ἤ A=C (B+Q) ἀντί γιά A=C*(B+Q). Καί τά δυό αὐτά παραδείγματα εἶναι λεγόμενα «συντακτικά λάθη». Ὁ ὑπολογιστής συχνά θά ὑποδείξει τή γραμμή τοῦ προγράμματος καθώς καί το εἶδος τοῦ λά- θους («SNTX ERROR AT 40» = συντακτικό λά- θος στή γραμμή 40), πράγμα πού εἶναι μεγάλη βοήθεια.

Ἕνα πρακτικό παράδειγμα διορθώσεως λάθους μέ τή βοήθεια ὑπολογιστῆ μπορεῖ νά δοθεῖ βά- ζοντας σκόπιμα λάθη σ' ἕνα ἀπό τά παραδείγμα- τα προγραμμάτων πού ἔχουν δοθεῖ στήν προη- γούμενη συνέχεια:

```
> 10 REM CALCULATION IF A!
> 20 LET N = 1
> 30 PRINT "ENTER A"
> 40 ILPUT A
> 50 IF A <= 1 THEN 90
> 60 LET N = N + X
> 70 NEXT X
> 80 PRINT A; "! ="; N
> 90 END
> RUN
ENTER A

SNTX ERROR AT 40
> 40 INPUT A
> RUN
ENTER A
? 3

NEXT ERROR AT 70
> 55 FOR X = 1 TO A
> RUN
ENTER A
? 3
3

CHAR ERROR AT 80
> 80 PRINT A, "! =", N
> RUN
ENTER A
? 3
3! = 7

BRK AT 90
>
```

Μετά ἀπό τήν πρώτη ἐντολή RUN, ὁ μεταφραστής ἄρχισε νά μεταφράζει καί νά ἐκτελεῖ τό πρόγραμμα. Στή γραμμή 40 βρέθηκε τό πρώτο λάθος (ILPUT ἀντί γιά INPUT) καί τύπωσε μιά προειδοποίηση. Μετά τή διόρθωση ἄρχισε πάλι ἡ ἐκτέλεση.

Ὅλα τώρα πᾶνε καλά μέχει τή γραμμή 70: Μετά τό NEXT δέν ὑπάρχει πρόταση FOR! Αὐτό προστίθεται στή γραμμή 55 καί ξαναρχίζει ἡ ἐκτέλεση. Ἀλλά τώρα κάτι συμβαίνει στή γραμμή 80. Ἡ ἄνω τελεία χρησιμοποιεῖται μόνο στό τέλος ὁλόκληρης πρότασης, τά ἐνδιάμεσα χωρίζονται μέ κόμμα.

Τώρα ἐπί τέλους τό πρόγραμμα ἐκτελεῖται μέχρι τό τέλος. Τά 'διαδικαστικά λάθη' βρέθηκαν ὅλα... ἀλλά τό ἀποτέλεσμα εἶναι λάθος! Τώρα πρέπει νά ριχτεῖ μιά πιό προσεκτική ματιά στό πρόγραμμα, ὅπως ἔχει ἀποθηκευτεῖ στή μνήμη τοῦ ὑπολογιστῆ:

```
> LIST
10 REM CALCULATION IF A!
20 LET N = 1
30 PRINT "ENTER A"
40 INPUT A
50 IF A <= 1 THEN 90
55 FOR X = 1 TO A
60 LET N = N + X
70 NEXT X
80 PRINT A, "! =", N
90 END

> 10 REM CALCULATION OF A!
> 60 LET N = N * X
> RUN
ENTER A
? 3
3! = 6

BRK AT 90
>
```

Ἐπί τέλους ἀνακαλύφθηκαν τά δύο τελευταῖα λάθη

α) ὀρθογραφικό λάθος στήν πρόταση REM – αὐτό δέν ἔχει ἐπίδραση στήν πρόταση REM, ἀλλά τό πρόγραμμα θά δείχνει καλύτερο ἄν διορθωθεῖ τό λάθος.

β) στή γραμμή 60 ἔχει εἰσαχθεῖ N+X ἀντί γιά N*X.

Σέ σύνθετα προγράμματα, τό «ξεκόλλημα» παίρνει πολλή περισσότερη ὥρα. Μερικές πρόσθετες προτάσεις PRINT μπορεῖ νά εἶναι πολύ βοηθητικές, τυπώνοντας ενδιαμεσα αποτελέσματα, ὥστε τό σημεῖο ὅπου βρίσκεται τό λάθος μπορεῖ νά ἐντοπιστεῖ γρηγορότερα. Π.χ., στό παραπάνω παράδειγμα '65 PRINT N, X' θά βοηθοῦσε πολύ στήν ἐντόπιση τοῦ λάθους. Ὅταν τό πρόγραμμα δουλεύει σωστά, οἱ περρισευούμενες προτάσεις μπορεῖ νά σβηστοῦν. Ἕνας ἄλλος τρόπος εἶναι νά ἐκτελεῖται μερικές φορές τό πρόγραμμα μέ διάφορες ἀρχικές τιμές. Ἔτσι φαίνεται τό εἶδος τοῦ λάθους πού συμβαίνει.

Τό ἀποτέλεσμα μερικῶν ἐνδιάμεσων ὑπολογισμῶν εἶναι αἰτία σφαλμάτων πολλές φορές. Αὐτό σέ μερικές περιπτώσεις διορθώνεται μέ μικρές τροποποιήσεις τῶν ὑπολογισμῶν. Π.χ. ἡ πρόταση: LET X = (C/(A-B)+B) * (A-B) γιά A=B θά ὁδηγήσει σέ λάθος διότι ὁ ὑπολογιστής δέν ἔχει τό «κοινό νοῦ» νά πολλαπλασιάσει πρῶτα μέ A-B καί θά διαιρέσει πρῶτα τό C μέ 0. Αὐτό τό πρόβλημα λύνεται ὡς ἑξῆς: LET X = C+B * (A-B).

Μερικές φορές εἶναι χρήσιμο νά σταματήσει ὁ ὑπολογιστής σέ κάποιο σημεῖο τοῦ προγράμματος. Ἕνας τρόπος νά γίνει αὐτό εἶναι νά προστεθεῖ μιά πρόταση INPUT: ὁ ὑπολογιστής θά σταματήσει σ' αὐτή τήν πρόταση, θά τυπώσει ἕνα '?' καί θά συνεχίσει μόλις ἕνας ἀριθμός εἰσαχτεῖ.

Τά ἐνδεικτικά λαθῶν ἔχουν ὡς ἑξῆς:

AREA ERROR = η μνήμη είναι πλήρης
CHAR ERROR = λάθος συμβολισμοῦ στή πρόταση
DIVO ERROR = διαίρεση μέ μηδέν
END ERROR = δέν ὑπάρχουν εἰσαγωγικά στό τέλος
FOR ERROR= FOR χωρίς NEXT
NEST ERROR = πολλά loops μέσα σέ ἄλλα loops
NEXT ERROR = NEXT χωρίς FOR.
NOGO ERROR = ὁ ἀριθμός γραμμῆς πού δείχνεται σέ πρόταση GOTO ἤ GOSUB δέν ὑπάρχει.
RTRN ERROR = RETURN χωρίς GOSUB
SNTX ERROR = συντακτικό λάθος
STMT ERROR = πρόταση χρησιμοποιημένη λάθος
UNTL ERROR = UNTIL χωρίς DO
VALU ERROR = 'λάθος ἀριθμός'

Ἐάν τά ἐνδεικτικά λαθῶν ἀναφέρονται σέ γραμμή προγράμματος, ὁ ἀριθμός τῆς γραμμῆς καθορίζεται, Π.χ. 'SNTX ERROR AT 30'.

Ἐκτεταμένο BASIC: - ἐπίσης γνωστό σάν προχωρημένο BASIC – εἶναι πιό εὐκολόχρηστη γλώσσα ἀπό τή BASIC. Μποροῦν τώρα πιά νά ἐξηγηθοῦν μερικές ἀπό τίς πρόσθετες εὐκολίες. Μερικές ἀπ' αὐτές εἶναι ἐπίσης γνωστές στή NIBL.

Μῆτρες: ἡ χρήση τους μπορεῖ νά ἁπλουστεύσει πολλές φορές ἐπιστημονικούς ὑπολογισμούς. Παράδειγμα μονοδιάστατης μήτρας, εἶναι A(1), A(2), A(3), A(4), A(5). Ἐδῶ ἔχουμε σειρά πέντε στοιχείων. Αὐτά εἶναι γνωστά σάν «στοιχεῖα» τῆς μήτρας. Ἕνα στοιχεῖο ἀντιπροσωπεύεται ἀπό ἕνα γράμμα, ἀκολουθούμενο ἀπό ἕνα ἀριθμό σέ παρένθεση. Μόνο ἀριθμοί μεταξύ 1 καί 10 χρησιμοποιοῦνται κανονικά, παρ' ὅλο πού σέ μερικές διαλέκτους BASIC μπορεῖ νά χρησιμοποιηθεῖ ἐπίσης τό 0. Εἶναι συχνά ἐπιτρεπτό νά περιλαμβάνεται μιά μεταβλητή ἤ ἄλλη ἔκφραση στήν παρένθεση, ἀντί γιά ἀριθμό: A(x), ἤ A(2+3). Ἡ χρήση μητρῶν φαίνεται παρακάτω:

```
10 REM EXAMPLE OF ARRAY
15 LET N = 0
20 FOR X = 1 TO 8
30 INPUT A(X)
40 N = N + A(X)
50 NEXT X
60 PRINT N
70 END
```

Σ' αὐτό τό πρόγραμμα τά στοιχεῖα A(1) ἕως A(B) προστίθενται σταδιακά.

Μιά μήτρα μπορεῖ νά ἐπεκταθεῖ σέ περισσότερες ἀπό μία «διαστάσεις». Μιά δισδιάστατη μήτρα, π.χ., μπορεῖ νά εἶναι ὡς ἀκολούθως:

A(1,1)	A(1,2)	A(1,3)	←σειρά
A(2,1)	A(2,2)	A(2,3)	
	↑ στήλη		

Ὅταν χρειάζονται πάνω ἀπό 10 στοιχεῖα σέ μονοδιάστατη σειρά ἤ πάνω ἀπό 10×10 σέ δισδιάστατη μήτρα, μέ πρόταση DIM μπορεῖ νά χρησιμοποιηθεῖ γιά νά ἐπεκτείνει τήν περιοχή. Π.χ. DIM A(50), B(20, 20) διαθέτει 50 μνῆμες γιά στοιχεῖα μονοδιάστατης μήτρας ('A') καί 20×20 (=400) γιά ('B'). Μιά πρόταση DIM μπαίνει κανονικά στήν ἀρχή προγράμματος.

Συχνά ἕνα τμῆμα προγράμματος χρησιμοποιεῖται πολλές φορές κατά τή διαδικασία προγράμματος. Μπορεῖ τότε νά μπαίνει σάν «πράξη». Ἕνα παράδειγμα:

```
> 10 REM USER DEFINED FUNCTION
> 20 DEF FNA(X,Y) = (X + Y) + (X * Y)
> 30 INPUT A, B
> 40 INPUT "FNA ="; FNA (A,B)
> 50 END
> RUN
? 3,4
FNA = 19
BRK AT 50
```

Σ' αὐτό τό παράδειγμα μιά πράξη καθορίζεται στή γραμμή 20. Μετά ἀπό τή λέξη κλειδί 'DEF' τό ὄνομα τῆς λειτουργίας καθορίζεται. Αὐτό περιλαμβάνει πάντοτε ἕνα FN ἀκολουθούμενο ἀπό ἕνα ἤ περισσότερα γράμματα. Μετά ἀπό τό ὄνομα τῆς λειτουργίας, FNA στό παραπάνω παράδειγμα, τοποθετοῦνται μεταβλητές σέ παρενθέσεις.

Ὅταν καθοριστεῖ αὐτή ἡ λειτουργία στό πρόγραμμα, ἐκτελοῦνται τά ἑπόμενα βήματα. Οἱ τιμές ἔχουν δοθεῖ καί ἐκτελεῖται ὁ ὑπολογισμός, ὅπως καθορίζεται στή γραμμή 20.

Ἡ ὁδηγία μπορεῖ νά δίνεται σέ περισσότερες ἀπό μία γραμμή:

```
10 DEF FNB(X,Y)
20 A= X + Y
30 B= X * Y
40 LET FNB= A + B
50 FNEND
60 PRINT FNB (3,4)
70 END
```

῞Οπως καί πρίν, ἡ λέξη DEF ἀκολουθεῖται ἀπό τό ὄνομα τῆς λειτουργίας καί τίς μεταβλητές (FNB(X,Y)). Μετά καθορίζεται ἡ πράξη, καί παίρνεται ἡ τελική τιμή.

Μία ἄλλη πρόσθετη λειτουργία, γνωστή καί στό σύστημα NIBL εἶναι ἡ λειτουργία RND, πού δίνει τυχαίους ἀριθμούς σέ περιοχή ἀπό 0,000000001 μέχρι 0,999999999. ῎Ετσι μπορεῖ νά παραχθεῖ τυχαῖος ἀριθμός ἀπό τό 1 ὡς τό 6, π.χ., ὡς ἑξῆς:

```
10 REM A= RND
20 LET A= RND
30 LET B= 6 * A + 1
40 LET C= INT B
50 PRINT C
60 END
```

Στή γραμμή 20, τό A παίρνει τυχαία τιμή μεταξύ 0 καί 1. Μετά τό B ἰσοῦται μέ τυχαῖο ἀριθμό μεταξύ 1 καί 7. ῎Ετσι τό C γίνεται τυχαῖος ἀριθμός μεταξύ 1 καί 6. Γιά εὐκρίνεια στό παραπάνω παράδειγμα οἱ διάφορες διαδικασίες ἁπλώνονται σέ ἀρκετές γραμμές. Πρακτικά, οἱ γραμμές 20 ἕως 40 θά ἐκτελοῦνται ταυτόχρονα: 20 LET C = INT (6*RND+1). Στή NIBL, πού παίρνει ἀκέραιους τυχαίους ἀριθμούς τό πρόγραμμα θά ἦταν ὡς ἑξῆς:

```
10 REM RND IN NIBL
20 C= RND(1, 6)
30 PRINT C
40 END
```

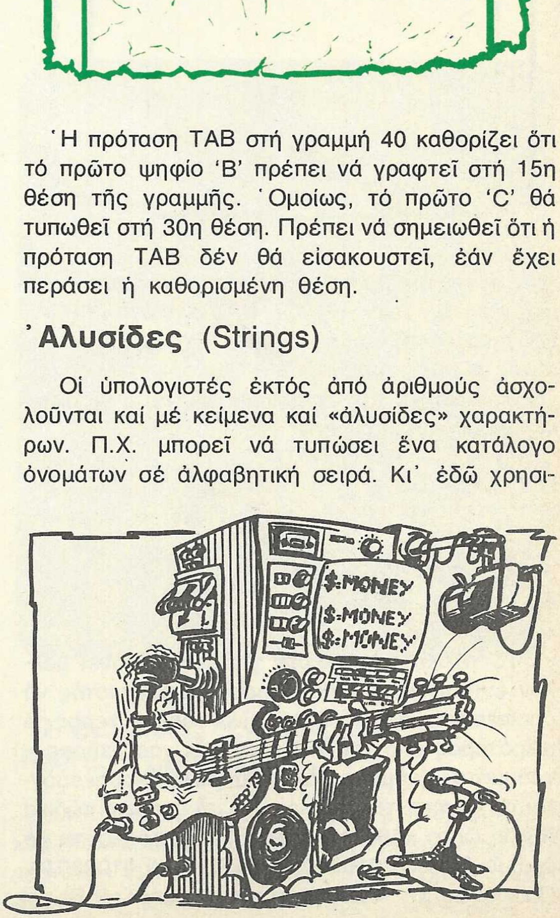
῾Η πρόταση TAB εἶναι ἐπέκταση τῆς πρότασης PRINT. Χρησιμοποιεῖται γιά νά καθορίσει σέ ποιά θέση σέ γραμμή ἕνας ἀριθμός ἤ φράση πρέπει νά ἀρχίσει. ῞Ενα παράδειγμα:

```
10 REM EXAMPLE OF TAB
20 INPUT A, B, C
30 PRINT A;
40 PRINT TAB(15), B;
50 PRINT TAB(30), C
60 END
```

῾Η πρόταση TAB στή γραμμή 40 καθορίζει ὅτι τό πρῶτο ψηφίο 'B' πρέπει νά γραφτεῖ στή 15η θέση τῆς γραμμῆς. ῾Ομοίως, τό πρῶτο 'C' θά τυπωθεῖ στή 30η θέση. Πρέπει νά σημειωθεῖ ὅτι ἡ πρόταση TAB δέν θά εἰσακουστεῖ, ἐάν ἔχει περάσει ἡ καθορισμένη θέση.

᾿Αλυσίδες (Strings)

Οἱ ὑπολογιστές ἐκτός ἀπό ἀριθμούς ἀσχολοῦνται καί μέ κείμενα καί «ἀλυσίδες» χαρακτήρων. Π.Χ. μπορεῖ νά τυπώσει ἕνα κατάλογο ὀνομάτων σέ ἀλφαβητική σειρά. Κι' ἐδῶ χρησι-



μοποιουνται μεταβλητές. Πρός διάκριση άπό ἀριθμητικές μεταβλητές, οἱ ἄλλες ἀκολουθοῦνται ἀπό εἰδικό σύμβολο – "$", π.χ. Τό μῆκος μιᾶς ἀλυσίδας συνήθως περιορίζεται στούς 15 χαρακτῆρες, περιλαμβανομένων καί τῶν κενῶν. "JOHN BULL" ἀποτελεῖται ἀπό 9 (καί τό κενό) κι ὄχι 8 χαρακτῆρες. Ἀκολουθεῖ ἕνα παράδειγμα προγράμματος ἀλυσίδας:

```
> 10 REM STRING HANDLING
> 20 READ A$, A, B$
> 30 PRINT A$, B$, A
> 40 DATA "ELEKTOR,", 50, "ISSUE"
> 50 END
> RUN
ELEKTOR, ISSUE 50
BRK AT 50
>
```

Στή γραμμή 20 ἡ ἀλφαριθμητική μεταβλητή A$ παίρνει τήν «τιμή» ΕΚΛΟΓΗ, τό A γίνεται 50 καί B$ παίρνει τή «τιμή» ISSUE. Στή γραμμή 30, αὐτές οἱ τιμές τυπώνονται σέ καθορισμένη σειρά. Ἡ δυνατότητα τῆς διαδικασίας φαίνεται σέ μερικά ἀκόμα παραδείγματα:

LET A$ = B$
LETZ A$ = "TOM" (ἄς προσεχτοῦν τά εἰσαγωγικά!)
JF A$ = B$ THEN...
IF A$<=B$ THEN...

Τό τελικό ἀποτέλεσμα μπορεῖ νά φανεῖ μᾶλλον ἐντυπωσιακό. Πῶς μπορεῖ ὁ ὑπολογιστής νά ἀποφασίσει, ἄν ὁ TOM εἶναι μεγαλύτερος ἤ μικρότερος ἀπό τόν TIM; Ὅλοι οἱ χαρακτῆρες – γράμματα, ἀριθμοί καί ἄλλα σύμβολα ἀποθηκεύονται στή μνήμη τοῦ ὑπολογιστῆ σέ δυαδικό κώδικα ASCII, ὥστε κάθε σύμβολο ν' ἀντιστοιχίζεται μέ ἀριθμό A=65 ἤ 01000001 καί B=66 ἤ 01000010. Προφανῶς τό A εἶναι μικρότερο ἀπό τό B.

PEEK POKE

Οἱ προτάσεις PEEK καί POKE χρησιμοποιοῦνται ἀπό μικροϋπολογιστές μέ γλώσσα BASIC. Ἡ πρόταση POKE ἀποθηκεύει ἕνα "byte" στή μνήμη. Ὁμοίως, μιά πρόταση PEEK ἀφαιρεῖ ἕνα "byte" ἀπό τή μνήμη. Ὁ τρόπος μέ τόν ὁποῖο ἀπευθύνεται κανείς στή μνήμη, ἐξαρτᾶται ἀπό τό ὑπό συζήτηση σύστημα τοῦ μικροϋπολογιστῆ. Π.χ. στό σύστημα NIBL χρησιμοποιεῖται ὁ συμβολισμός @ κάνοντας ἁπλή τήν ἀναφορά σέ προτάσεις PEEK καί POKE.

Λογικοί ἐκτελεστές

Μιά «λογική ἐκτέλεση», εἶναι καλά καθορισμένη χρήση δυαδικῶν ἀριθμῶν. Πλήρης περιγραφή εἶναι ἔξω ἀπό τά ὅρια αὐτοῦ τοῦ κειμένου. Οἱ πιό πολλές διάλεκτοι ἀναγνωρίζουν τίς λογικές λειτουργίες OR, AND καί NOT. Ὅλες οἱ ἄλλες λειτουργίες σχηματίζονται συνδυάζοντας αὐτές τίς τρεῖς. Μερικά παραδείγματα:

IF(x=1 AND Y=1) THEN...

Ἄν ἱκανοποιοῦνται καί οἱ δύο συνθῆκες (x=1 καί y=1), ἐκτελεῖται ἡ ὁδηγία πού ἀκολουθεῖ τό THEN.

Y=NOT X

Στό Y ἀνατίθεται τό ἀντίθετο τοῦ X. Ἐπειδή οἱ λογικές ἐκτελέσεις ἀναφέρονται σέ δεκαεξαδικό σύστημα ἀριθμῶν, τό ἀντίθετο εἶναι παράξενο. Π.χ. τό ἀντίθετο τοῦ 3 (=0011) εἶναι 12(=1100).

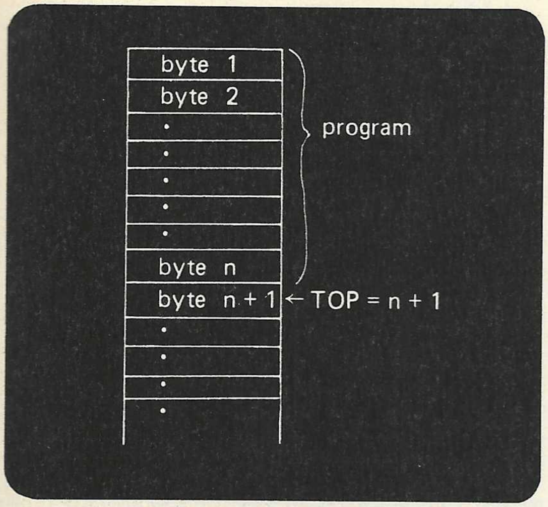
Ἡ σειρά τῶν ἐκτελέσεων ἔχει ὡς ἀκολούθως: τό OR ἔχει τήν ἴδια προτεραιότητα μέ τό + καί –, τό AND ἔχει τήν ἴδια προτεραιότητα μέ τό * καί /, καί τό NOT ἔχει τή μέγιστη προτεραιότητα.

Προτάσεις NIBL καί δυνατότητες

α) Ἡ λειτουργία MOD φαίνεται γιά λειτουργία μέ τήν ὁποία τό σύστημα NIBL περιλαμβάνει κλάσματα. Κατά κάποιο τρόπο, MOD (X,Y) ὑπολογίζει τήν ἀπόλυτη τιμή τοῦ ὑπόλοιπου ἀπό τή διαίρεση X/Y. Μερικά παραδείγματα: $14/3=4^2/_3$. Τό ὑπόλοιπο εἶναι 2, ἡ ἀπόλυτη τιμή τοῦ ὁποίου εἶναι 2, ὥστε MOD (14,3) ἰσοῦται μέ 2.
$-25/7=-3^4/_7$.

Τό ὑπόλοιπο εἶναι -4, ἡ ἀπόλυτη τιμή τοῦ -4 εἶναι 4, ἔτσι MOD (-25,7) ἰσοῦται μέ 4. Ἔτσι, ἡ πρόταση Y=MOD(3,4) κάνει τή μεταβλητή Y ἴση μέ 3.

Ἡ λειτουργία TOP βρίσκει τήν πρώτη ἀχρησιμοποίητη θέση στή μνήμη. Αὐτό ἰσοῦται μέ τήν «κορυφή» τῆς ἀχρησιμοποίητης περιοχῆς τῆς



μνήμης:

Αὐτό εἶναι χρήσιμο ὅταν χρειάζεται κενή θέση - ὅταν δηλαδή ὁ ἐκτελεστής θέλει ν' ἀποθηκεύσει πληροφορίες σέ συγκεκριμένη θέση τῆς μνήμης.

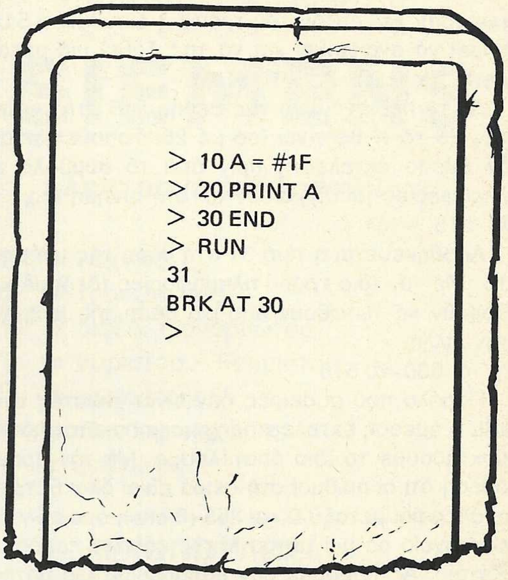
Ψευδομεταβλητές: Δύο ψευδομεταβλητές εἶναι γνωστές στό NIBL: PAGE καί STAT. Καί οἱ δύο μπαίνουν σ' ὁποιοδήποτε μέρος ἑνός σήματος ἴσον: "LET Y = PAGE" κάνει τό Y ἴσον μέ τόν ἀριθμό τῆς σελίδας (page number). "LET PAGE = Y" κάνει τόν ἀριθμό τῆς σελίδας ἴσον μέ τό Y.

Ἡ ψευδομεταβλητή STAT μπορεῖ νά χρησιμοποιηθεῖ γιά νά ζητήσει μιά ἀνατύπωση τοῦ περιεχομένου τοῦ καταχωρητῆ: PRINT STAT. Ἐπίσης μπορεῖ νά χρησιμοποιηθεῖ γιά νά θέσει τόν καταχωρητή σέ μιά ἐπιθυμητή τιμή: STAT=15.

Ὅπως συνήθως, ἡ τιμή μετά τό ἴσον μπορεῖ νά εἶναι ἕνας ἀριθμός, μιά μεταβλητή ἤ μιά ἔκφραση. Αὐτό πρῶτα μετατρέπεται σέ ἁπλό δυαδικό ἀριθμό καί μετά ἀποθηκεύεται στό καταχωρητή. Προφανῶς μόνο ἕνα byte (8 bits) μπορεῖ νά ἀποθηκευτεῖ στόν καταχωρητή. Χρησιμοποιεῖται byte χαμηλῆς τάξεως (πλέον ἀσήμαντα bits) τό «ὑψηλό» byte ἀγνοεῖται.

Τό βασικό πλεονέκτημα τοῦ STAT εἶναι ὅτι δίνει ἄμεση προτεραιότητα στίς γραμμές I/O στό SC/MP τσίπ.

Δεκαεξαδικοί ἀριθμοί: Μόνο ἀκέραιοι ἀριθμοί ἀναγνωρίζονται στό NIBL. Μέχρι τώρα ὑποθέσαμε ὅτι αὐτοί πρέπει νά εἶναι δεκαδικοί ἀριθμοί. Τό NIBL, ὡστόσο, μπορεῖ νά λειτουργήσει μέ δεκαεξαδικούς ἀριθμούς. Ἕνας δεκαεξαδικός ἀριθμός συνοδεύεται ἀπό τό σῆμα #. Π.χ:



δεκαδικό σύστημα	δεκαεξαδικό σύστημα
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
17	11
18	12
κ.τ.λ.	κ.τ.λ.

Ὁ ἔμμεσος ἐκτελεστής indirect operator

Στό NIBL οἱ προτάσεις PEEK καί POKE ἀντικαθίστανται ἀπό τόν ἔμεσο ἐκτελεστή «indirect operator» @. Τό σύμβολο @ ἀκολουθεῖται ἀπό μία διεύθυνση (δίδεται σάν ἀριθμός, μεταβλητή ἤ

ἔκφραση). Μ᾽ αὐτόν τόν τρόπο ἡ διεύθυνση 515 μπορεῖ νά ἀνακληθεῖ καί νά τῆς δοθεῖ μιά μεταβλητή π.χ. LET X = @ 515.

῞Αν τό περιεχόμενο τῆς θέσης 515 στή μνήμη ἦταν 25 τό Χ θά γίνει ἴσο μέ 25. Τοποθετώντας τόν ἔμεσο ἐκτελεστή πρίν ἀπό τό σύμβολο = πληροφορίες ἀποθηκεύονται στή μνήμη π.χ.

@ 515 = 31

᾽Αποθηκεύεται ἡ τιμή 31 στή θέση τῆς μνήμης 515. Μέ τόν ἴδιο τρόπο πληροφορίες (δεδομένα) μποροῦν νά τυπωθοῦν ἀπό μιά θέση τῆς μνήμης στήν ἄλλη.

π.χ. @ 530=@ 515

Παρόλο πού οἱ σειρές δέν εἶναι γνωστές στό NIBL ὁ ἔμεσος ἐκτελεστής χρησιμοποιεῖται γιά νά ἀποκτήσουμε τό ἴδιο ἀποτέλεσμα. Μέ τήν προϋπόθεση ὅτι οἱ ἀριθμοί στή σειρά εἶναι ὅλοι θετικοί καί ἀκέραιοι μεταξύ 0 καί 255 (8 bits), ὁποιοδήποτε στοιχεῖο σέ μιά μήτρα M×N, εὔκολα προσδιορίζεται. ῞Ας πάρουμε σάν παράδειγμα μιά μήτρα 4×5 (M=4 N=5).

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4 ←Row I
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4

M γραμμές

↑ Column J

N. στῆλες

῞Αν θέλουμε νά ἀποθηκεύσουμε αὐτή τή μήτρα στή μνήμη ἀπό τή διεύθυνση Α (καί μέ τήν προϋπόθεση ὅτι ὑπάρχει ἀρκετός χῶρος κάτω ἀπό τό Α γιά νά χωρέσει) τά στοιχεῖα θά ἀποθηκευθοῦν στίς ἑξῆς διευθύνσεις

ΔΙΕΥΘΥΝΣΗ	ΣΤΟΙΧΕΙΟ
Α	0,0
Α + 1	0,1
Α + 2	0,2
Α + 3	0,3
Α + 4	0,4
Α + 5	1,0
Α + 6	1,1
Α + 7	1,2
Α + 8	1,3
κ.τ.λ	κ.τ.λ

Τό στοιχεῖο (1,2) π.χ. ἀποθηκεύεται στή διεύθυνση A+1×5+2=A+7. Γενικά ἡ θέση ὁποιουδήποτε στοιχείου (I,J) σέ μιά μήτρα (M,N) καθορίζεται σάν @ (A+I*N+J). Βεβαίως αὐτό εἶναι πιό κουραστικό ἀπό τό νά καθορίζεται τό A(I,J), ἀλλά μᾶς παρέχει ἀποτέλεσμα.

Δημιουργία ἁλυσίδας

Οἱ ἱκανότητες πού ἔχει τό NIBL νά δημιουργεῖ ἁλυσίδες, εἶναι μᾶλλον περιορισμένες σχετικά μέ τίς ἄλλες διαλέκτους τῆς BASIC καί μάλιστα τελείως διαφορετικές ἀπ᾽ αὐτές πού περιγράψαμε προηγουμένως. Π.χ. ἡ πρόταση: 80 INPUT $6150, ὅταν παρουσιάζεται σ᾽ ἕνα πρόγραμμα ἔχει σάν ἀποτέλεσμα ὁ ὑπολογιστής νά ἀνταποκριθεῖ τυπώνοντας ἕνα ἐρωτηματικό, ὅπως συμβαίνει καί μέ μιά ἁπλή INPUT πρόταση. Μιά ἁλυσίδα (πού ἀποτελεῖται ἀπό σειρά χαρακτήρων), μπορεῖ τώρα νά εἰσαχτεῖ ἀκολουθούμενη ἀπό τό CR. Αὐτή ἡ ἁλυσίδα χαρακτήρων ἀποθηκεύεται σέ διαφορετικές θέσεις τῆς μνήμης πού ἀρχίζουν στήν καθορισμένη διεύθυνση (τό παράδειγμα ἀπό 6150). Δέν ἀπαιτοῦνται εἰσαγωγικά, διότι θά ἀποθηκεύονταν σάν μέρος τῆς ἁλυσίδας. ᾽Ακόμα καί τό τελικό CR ἀποθηκεύεται στό τέλος τῆς ἁλυσίδας. Εἶναι ἐπίσης δυνατό νά καθοριστεῖ μιά διεύθυνση στό δεκαεξαδικό:

INPUT $ # 180A

᾽Αφοῦ ἀποθηκευτεῖ μιά ἁλυσίδα, ἄλλες, διαδικασίες, ἔλεγχοι κ.λπ, μποροῦν νά διεκπεραιωθοῦν μέσω τοῦ ἔμεσου ἐκτελεστῆ. Εἶναι ἐπίσης δυνατόν νά μεταφερθεῖ ἡ ἁλυσίδα ἀπό μία σειρά τῶν θέσεων τῆς μνήμης σέ μία ἄλλη:

$ ΔΙΕΥΘΥΝΣΗ ΠΡΟΟΡΙΣΜΟΥ = $ ΤΩΡΙΝΗ ΘΕΣΗ.

Αὐτό κάνει ὅλους τούς χαρακτῆρες στήν ἁλυσίδα νά τυπωθοῦν ἕνας ἕνας, ἀπό τήν τωρινή θέση σέ θέσεις τῆς μνήμης πού ἀρχίζει ἀπό τήν διεύθυνση ὅπου προορίζονται π.χ. $515=360

Αὐτή ἡ ἐντολή θά κάνει ἀντιμετάθεση τῶν συμβόλων τῆς ἁλυσίδας ὡς ἑξῆς: Τό περιεχόμενο τῆς μνήμης 360 θά τοποθετηθεῖ σέ θέσεις ἀρχίζοντας ἀπό τήν θέση 515. Τό CR ἐπίσης τυπώνεται καί ἀναγνωρίζεται σάν τό τέλος τῆς ἁλυσίδας. ῞Ας σημειωθεῖ ὅτι ἡ σειρά διευθύνσεων προορισμοῦ δέν πρέπει νά ὑπερκαλύπτει τίς θέσεις τίς τωρινές πού βρίσκεται ἡ ἁλυσίδα. ῎Ετσι στό προηγούμενο παράδειγμα οἱ νέες διευθύνσεις μπορεῖ νά ἔχουν τιμές ἀπό 360 ἕως καί 514, ὄχι 515. Αὐτό μπορεῖ νά ἔχει καταστροφικά ἀποτελέσματα καί νά σβηστεῖ ἡ μνήμη.

ΕΠΙΛΟΓΟΣ

Στίς τέσσερεις αὐτές συνέχειες προσπαθήσαμε νά καλύψουμε σφαιρικά τό τεράστιο θέμα πού λέγεται «BASIC». Τό πόσο τό καταφέραμε θά τό κρίνετε ἐσεῖς. Αὐτό πού πρέπει νά τονίσουμε εἶναι ἡ δυναμικά ἐξελισσόμενη τεχνολογία τῶν μικροϋπολογιστῶν. Ἀργά ἤ γρήγορα θά μποῦν στήν ζωή μας σάν ἕνα ἀπαραίτητο βοήθημα σ' ὅλες τίς πράξεις μας. Στόν Ἑλλαδικό χῶρο γενικά λείπει τό θεωρητικό ὑπόβαθρο γιά τό θέμα αὐτό. Προσφέροντας αὐτές τίς γνώσεις γιά τήν γλώσσα προγραμματισμοῦ, πιστεύουμε ὅτι δώσαμε μιά καλή ἀφορμή γιά παραπέρα μελέτη καί ἐμβάθυνση στό θέμα.

Ζητᾶμε κατανόηση ἄν ὑπῆρξαν λάθη στήν προσπάθεια πού κάναμε γιά τήν κατάλληλη ἀπόδοση τῶν ξένων ὅρων (πού δημιουργήθηκαν μέ τήν νέα αὐτή τεχνολογία) στά ἑλληνικά, μιά καί δέν ὑπάρχει ἀκόμα καθιερωμένη σχετική ἑλληνική ὁρολογία. Πάντως γιά ὅσους ἐνδιαφέρονται παραθέτουμε μιά σύντομη βιβλιογραφία γιά παραπέρα μελέτη.

ΜΙΧΑΛΗΣ ΟΙΚΟΝΟΜΑΚΗΣ.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) **CHU Y.:** Computer Organization and Microprogramming» Prentice-Hall 1972.
2) **HUSSON S.:** Microprogramming Principles and Practices» Prentice-Hall 1975.
3) **ELEKTOR:** «BASIC» (περιοδικό).
4) **INTEL corp.:** «A Guide to PL/M Programming» 1973.

καταστημα προϊοντων

SIEMENS

ἠλεκτρονικά ἐξαρτήματα

- λυχνίες
- τρανζίστορς
- δίοδοι - ἀνορθωτές
- θυρίστορ - θερμίστορ
- φωτοστοιχεῖα
- ὁλοκληρωμένα IC
- φερρίτες

Τ. ΜΙΧΟΠΟΥΛΟΣ
ΜΑΡΝΗ 38 - ΑΘΗΝΑ
ΤΗΛ. 5225122–5230275

ΦΡΕΙΔΕΡΙΚΟΣ ΓΑΛΑΝΟΠΟΥΛΟΣ
ΚΤΗΜΑΤΟΜΕΣΙΤΗΣ

σᾶς ἐνδιαφέρει ἡ σωστή ἐπένδυση καί ἡ σωστή τοποθέτηση χρημάτων; ἄν ναί, ἐλάτε γιά μιά φιλική-συζήτηση.

Ἐμεῖς σᾶς περιμένουμε καί σᾶς προσφέρουμε:

- ○ αὐτοτελῆ κεντρικά ἀκίνητα
- ○ κεντρικά καταστήματα
- ○ θέσεις γιά πάρκινγκ (στύλ γκαρσονιέρας)
- ○ βίλλες ὅλων τῶν κατηγοριῶν
- ○ δάνεια πολλῶν ἑκατομμυρίων κ.λ.π.

Τηλεφωνεῖστε μας στά:
τηλ. 5224233 – 5240743 (5-9 μ.μ.)